Oscar R. Frost
PHP &
MySQL
Guida completa alla programmazione
web lato server e database SQL
CONTIENE ESEMPI DI CODICE ED ESERCIZI PRATICI
MySQL
PHP

&
P
P
M

PHP e MYSQL

PHP e MYSQL

PHP

Premessa

Capitolo 1 Cos'è PHP

Capitolo 2 Strumenti

Capitolo 3 Installazione

Windows
MacOS X
Dove sono i file?
Verificare l'installazione
Capitolo 4 Codice PHP

Variabili
Gli array
Valori speciali
Istruzioni condizionali
Mostrare risultati
Capitolo 5 Tipi di dati

Capitolo 6 I cicli

While e do…while
For e foreach
Capitolo 7 Funzioni

Require e include
Conclusioni

MySQL

Premessa

Capitolo 1 Cos'è un DB

Capitolo 2 Punti di forza

Capitolo 3 Installazione

Linux

Windows
macOS
Capitolo 4 Comandi SQL
Database
Tabelle
CREATE
INSERT
Chiavi primarie
ALTER TABLE
DROP TABLE
DROP DATABASE
Capitolo 5 Recuperare i dati
SELECT
DISTINCT
Capitolo 6 Gli operatori
Aritmetici
Confronto
Caratteri
Logici
Insiemistici
Capitolo 7 Funzioni
Funzioni aggregate
Funzioni per data e ora
Funzioni aritmetiche
Funzioni per caratteri

Capitolo 8 JOIN delle tabelle

INNER JOIN

LEFT JOIN

RIGHT JOIN

SELF JOIN

Conclusioni

PHP

Premessa

Durante il corso degli anni diversi linguaggi si sono susseguiti, alcuni sono stati usati per poco tempo, altri sono più longevi. PHP è sicuramente un linguaggio longevo poiché nel 1994 nasceva la sua prima versione. Nel corso degli anni molti altri linguaggi hanno cercato di prendere il suo posto, alcuni si sono avvicinati, altri non ci sono riusciti tuttavia PHP è un linguaggio che ha continuato ad evolversi per restare al passo con i tempi. In questo modo ha facilitato lo sviluppo ai programmatori Web aggiungendo funzionalità sempre nuove ed interessanti, ottimizzate in modo davvero eccellente.

PHP non è un linguaggio difficile, ma non è nemmeno molto semplice infatti ogni sito Web è diverso, quindi è impossibile prendere uno script, incollarlo in una pagina Web e aspettarsi che funzioni. L'obiettivo è quello di aiutare i web designer con poca o nessuna conoscenza della programmazione a guadagnare la fiducia necessaria per immergersi nel codice e adattarlo alle proprie esigenze.

Capitolo 1
Cos’è PHP

Ufficialmente, PHP sta per PHP: Hypertext Preprocessor e, anche se può sembrare un nome poco carino, questo libro ha lo scopo di aiutarti a programmare con PHP e nel frattempo farti capire cosa rende così felici i programmatori PHP e perché lo usano. PHP è un linguaggio di scripting che dà vita ai siti Web in diversi modi:

- Carica i file attraverso una pagina Web
- Genera miniature da immagini più grandi
- Legge e scrive su file
- Visualizza e aggiorna le informazioni in modo dinamico
- Può usare un database per visualizzare e archiviare informazioni
- Rende indicizzabili i siti Web
- E tanto altro ancora…

Leggendo questo ebook, sarai in grado di fare tutto questo. PHP è facile da imparare; è indipendente dalla piattaforma, quindi lo stesso codice funziona sia su Windows, che su Mac OS X che su Linux e tutto il software che devi sviluppare con PHP è open source quindi gratuito. PHP è una tra le tecnologie più utilizzate per la creazione di siti Web dinamici, ma uno dei suoi obiettivi principali era quello di creare un libro degli ospiti raccogliendo informazioni da un modulo online e visualizzandolo su una pagina web.

PHP viene utilizzato per creare contenuti dinamici da oltre l'80 percento dei 10 milioni di siti Web che analizza regolarmente. È il linguaggio che muove i più famosi sistemi di gestione dei contenuti (CMS) come Drupal, Joomla! e WordPress. Oltre a questi viene usato anche da Facebook e Wikipedia. Già questo basterebbe per dare fiducia ad un linguaggio ma una delle grandi attrazioni di PHP, tuttavia, è che rimane fedele alle sue radici. Il creatore originale di PHP, Rasmus Lerdorf, lo descrisse come "un linguaggio di scripting molto adatto ai programmatori, adatto a persone con poca o

nessuna esperienza di programmazione così come allo sviluppatore Web esperto che ha bisogno di risolvere problemi rapidamente." Puoi iniziare a scrivere script interessanti senza dover imparare molta teoria, ma sai che la stessa tecnologia può essere usata anche a livello industriale o per progetti molto grandi.

PHP è stato originariamente progettato per essere incorporato nell'HTML di una pagina Web, ed è così che viene spesso utilizzato. Ad esempio, se si desidera visualizzare l'anno corrente in un avviso di copyright, è possibile inserirlo nel piè di pagina:

```
<p> &copy; <?php echo date ( 'Y' ); ?> Copyright </p>
```

Su un server Web abilitato per PHP, il codice tra i tag <?php e ?> viene elaborato automaticamente e visualizza l'anno corrente in questo modo:

© 2020 Copyright

Questo è solo un esempio banale, ma illustra alcuni dei vantaggi dell'utilizzo di PHP:

- ogni utente vedrà l'anno corretto perché è il server che lo imposta;
- la data viene calcolata dal server Web, quindi non viene influenzata dall'orologio del computer dell'utente che potrebbe essere impostato in modo errato.

Sebbene sia conveniente incorporare il codice PHP in HTML in questo modo, è ripetitivo e può causare errori. Può anche rendere difficile la gestione delle tue pagine Web, in particolare quando inizi a utilizzare un codice PHP più complesso. Di conseguenza, è buona norma memorizzare il codice in file separati e quindi utilizzare PHP per creare le tue pagine dai diversi componenti.

Questi file separati possono contenere solo PHP, solo HTML o una combinazione di entrambi creando delle vere e proprie componenti riusabili.

A titolo di esempio, puoi inserire il menu di navigazione del tuo sito Web in un file da includere e utilizzare PHP per includerlo in ogni pagina. Ogni volta che è necessario apportare modifiche al menu, si modifica solo un file e le modifiche si riflettono automaticamente in ogni pagina che include il menu. Immagina quanto tempo puoi risparmiare in un sito Web con centinaia di pagine!

Con una normale pagina HTML, il contenuto viene fissato dallo sviluppatore Web in fase di progettazione e caricato sul server Web. Quando qualcuno visita la pagina, il server Web invia semplicemente l'HTML e altre risorse, come le immagini e i fogli di stile. È un’architettura molto semplice: la richiesta proviene dal browser e il contenuto, impostato dal programmatore, viene rispedito dal server. Quando crei pagine Web con PHP, non è così semplice infatti quando viene visitato un sito Web basato su PHP, viene attivata la seguente sequenza di eventi:

1. Il browser invia una richiesta al server web;

2. Il web server passa la richiesta al motore PHP, che è incorporato nel server;

3. Il motore PHP elabora il codice. In molti casi, vengono eseguite anche delle query su un database;

4. Il server invia la pagina completa al browser.

Questo processo di solito richiede solo qualche frazione di secondo, quindi è improbabile che il visitatore di un sito Web PHP noti alcun ritardo. Poiché ogni pagina è costruita individualmente, i siti PHP possono rispondere all'input dell'utente, visualizzando contenuti specifici, ad esempio, dopo aver effettuato l’accesso al sito.

PHP è una lingua lato server ed il codice PHP rimane sul server web. Dopo che è stato elaborato, il server invia solo l'output dello script al browser del visitatore. Normalmente si tratta di HTML, ma

PHP può essere utilizzato anche per generare altri linguaggi Web, come JSON (JavaScript Object Notation). PHP ti consente di introdurre una logica nelle tue pagine web basata su alternative. Alcune decisioni vengono prese utilizzando le informazioni che PHP raccoglie dal server: la data, l'ora, il giorno della settimana, le informazioni nell'URL della pagina e così via. Immaginiamo di creare un sito Web con i programmi TV del giorno corrente, siamo sicuri che se è mercoledì, mostrerà i programmi TV del mercoledì. Altre volte, le decisioni si basano sull'input dell'utente, che PHP estrae dai moduli online. Se ti sei registrato ad un sito, verranno visualizzate informazioni personalizzate come nome, cognome, immagine del profilo ecc. Tutto questo è sicuro perché il sito Web riceve solo l’output di uno script ma non sa come è fatto lo script.

Riguardo la sicurezza, PHP è come l'elettricità o i coltelli da cucina della tua casa: gestiti correttamente, sono molto sicuri; gestiti in modo irresponsabile, possono causare molti danni. PHP non è pericoloso, né tutti devono diventare esperti di sicurezza per utilizzarlo. L'importante è comprendere il principio di base della sicurezza PHP: controllare sempre l'input dell'utente prima di elaborarlo. La maggior parte dei rischi per la sicurezza può essere eliminata con il minimo sforzo. Il modo migliore per proteggerti è capire il codice che stai utilizzando e metterti nei panni di un utente malevolo che vuole rubare dati dal tuo server.

Capitolo 2
Strumenti

Per essere precisi, non è necessario alcun software speciale per scrivere script PHP infatti il codice PHP è un semplice testo e può essere creato in qualsiasi editor di testo, come Blocco note su Windows o TextEdit su Mac OS X. Detto questo, programmare sarà molto più semplice se usi un software con funzionalità progettate per accelerare la processo di sviluppo. Ce ne sono molti disponibili, sia gratuiti che a pagamento.

Grazie a questi strumenti se c'è un errore nel tuo codice, te ne potrai accorgere ben prima di arrivare al browser bensì direttamente in fase di programmazione e non in fase di test. Dovresti scegliere un editor con le seguenti caratteristiche:

- Controllo della sintassi PHP: i correttori di sintassi monitorano il codice durante la digitazione e evidenziano subito gli errori, risparmiando molto tempo e frustrazioni;
- Colorazione della sintassi PHP: il codice viene evidenziato in diversi colori in base al ruolo che svolge. Se il tuo codice ha un colore rosso o inaspettato, molto probabilmente hai commesso un errore;
- Suggerimenti sul codice PHP: PHP ha così tante funzioni integrate che può essere difficile ricordare come usarle, anche per un utente esperto. Molti editor di script visualizzano automaticamente i suggerimenti con promemoria su come funziona o sui parametri in ingresso;
- Numerazione delle righe: trovare rapidamente una riga specifica semplifica notevolmente la risoluzione dei problemi;
- Una funzione per bilanciare le parentesi: parentesi (()), parentesi quadre ([]) e parentesi graffe ({}) devono sempre trovarsi in coppie corrispondenti. È facile

dimenticare di chiudere un paio, tuttavia, gli editor di script ti aiutano a risolvere questo problema.

Anche se non hai intenzione di sviluppare molto codice in PHP, dovresti prendere in considerazione l'uso di un editor dedicato se il tuo programma di sviluppo Web non supporta il controllo della sintassi. I seguenti editor dedicati hanno tutte le funzionalità essenziali, come il controllo della sintassi e suggerimenti sul codice. Non è un elenco esaustivo, ma piuttosto basato sull'esperienza personale.

- PhpStorm è programma di editing PHP dedicato ed offre un eccellente supporto per HTML, CSS e JavaScript. Attualmente è il mio programma preferito per lo sviluppo con PHP.
- Sublime: se sei un fan di Sublime Text, ci sono plug-in per la colorazione della sintassi PHP, il controllo della sintassi e la documentazione.
- Zend Studio: se sei davvero serio sullo sviluppo di PHP, Zend Studio è l'ambiente di sviluppo integrato (IDE) più completo per PHP. Zend Studio funziona su Windows, Mac OS X e Linux. In passato era costoso, ma il prezzo per i singoli sviluppatori è ora molto più conveniente.
- Atom: è un IDE davvero personalizzabile, esiste un plugin per tutto, PHP incluso.
- Komodo: IDE gratuito e open source per PHP e una serie di altri linguaggi informatici popolari. È disponibile per Windows, Mac OS X e Linux.

Allora configura il tuo ambiente perché ora siamo pronti per partire ed iniziare a sviluppare.

Capitolo 3
Installazione

Ora che hai deciso di utilizzare PHP per arricchire le tue pagine web, devi assicurarti di avere tutto il necessario per andare avanti. Anche se puoi testare tutto sul tuo server remoto, di solito è più conveniente testare le pagine PHP sul tuo computer locale tanto tutto ciò che devi installare è gratuito. In questo capitolo, spiegherò le varie opzioni per Windows e Mac OS X. I componenti necessari sono normalmente installati di default su Linux.

Il modo più semplice per scoprire se il tuo sito Web supporta PHP è chiedere alla tua società di hosting ma solitamente PHP è supportato di default. L'altro modo per scoprirlo è caricare una pagina PHP sul tuo sito Web e vedere se funziona. Anche se sai già che il tuo sito supporta PHP, fai il seguente test per vedere quale versione è in esecuzione:

1. Aprire un editor di testo, ad esempio Blocco note o TextEdit, e digitare il seguente codice in una pagina vuota:

```
<? php echo phpversion (); ?>
```

2. Salvare il file come *phpversion.php* . È importante assicurarsi che il sistema operativo in uso non aggiunga l'estensione *.txt* dopo *.php* . Gli utenti Mac dovrebbero inoltre assicurarsi che TextEdit non salvi il file in formato RTF (Rich Text Format);
3. Carica *phpversion.php* sul tuo sito web come faresti con una pagina HTML, quindi digita l'URL in un browser. Supponendo che tu abbia caricato il file al livello più alto del tuo sito, l'URL sarà simile a http://www.tuosito.it/phpversion.php . Se sullo schermo vedi un numero diviso in tre parti come 5.6.1 PHP è abilitato. Il numero indica quale versione di PHP è in

esecuzione sul tuo server. È necessario disporre almeno della versione 5.4.0 per le funzioni di questo ebook;

4. Se ricevi un messaggio di errore probabilmente PHP è supportato ma hai commesso un errore nel digitare il codice nel file;
5. Se vedi solo il codice che hai inserito senza alcuna versione di PHP, significa che PHP non è supportato.

Se il tuo server esegue PHP 5.3 o precedente, contattalo per informarlo che desideri la versione più recente di PHP.

A differenza delle normali pagine Web, non puoi semplicemente fare doppio clic sulle pagine PHP e visualizzarle nel tuo browser. Queste pagine devono essere analizzate o elaborate attraverso un server Web che supporta PHP. Se la tua società di hosting supporta PHP, puoi caricare i tuoi file sul tuo sito Web e testarli in remoto, tuttavia, devi caricare il file ogni volta che apporti una modifica. Se vuoi lavorare subito con PHP, usa il tuo sito web come banco di prova. Scoprirai presto la necessità di un ambiente di test PHP in locale quindi vediamo come fare.

Per testare le pagine PHP sul tuo computer locale, devi installare quanto segue:

- Un server Web: un software che visualizza pagine Web che può risiedere nel tuo stesso pc;
- PHP;
- MySQL e un front-end per MySQL chiamato *phpMyAdmin* , necessari per funzionare con un database.

Per molti anni, ho raccomandato di installare separatamente ogni componente di un ambiente di test PHP, piuttosto che utilizzare un pacchetto contenente *Apache* , *PHP* , *MySQL* e *phpMyAdmin* in un'unica operazione. All'inizio erano facili da installare ma erano quasi impossibili da disinstallare o aggiornare. I pacchetti all-in-one attualmente disponibili sono eccellenti e non esito a raccomandarli

ora infatti sui miei computer, utilizzo **XAMPP** per Windows e **MAMP** per Mac OS X. Sono disponibili altri pacchetti ma non importa quale scegli.

Windows

Per impostazione predefinita, la maggior parte dei computer Windows nasconde l'estensione di tre o quattro lettere, come *.doc* o *.html* , quindi tutto ciò che vedi nelle finestre di dialogo così come in Esplora file di Windows nasconde la sua estensione. Windows 8 visualizza l'estensione del file per i file PHP, ma è utile attivare la visualizzazione dell'estensione del nome per tutti i file. Utilizza queste istruzioni per abilitare la visualizzazione delle estensioni di file in Windows 8/10:

1. Apri Esplora file.
2. Seleziona Visualizza per espandere la barra multifunzione nella parte superiore della finestra Esplora file.
3. Selezionare la casella di controllo "Estensioni nome file".

Visualizzando l’estensione dei file sei più al sicuro: puoi sapere se un documento dall'aspetto innocente è in realtà un virus.

La maggior parte delle installazioni PHP vengono eseguite sul server Web Apache. Entrambi sono open source e funzionano bene insieme. Tuttavia, Windows ha un proprio server Web, Internet Information Services (IIS), che supporta anche PHP. Microsoft ha lavorato a stretto contatto con il team di sviluppo di PHP per migliorare le prestazioni di PHP su IIS eguagliando Apache. Quindi, quale dovresti scegliere? La risposta dipende dal fatto che si sviluppino pagine Web utilizzando ASP o ASP.NET o che si intenda farlo. A meno che tu non abbia bisogno di IIS per ASP o ASP.NET, ti consiglio di installare Apache, usando XAMPP o uno degli altri pacchetti all-in-one. Se è necessario utilizzare IIS, il modo più conveniente per installare PHP è utilizzare il programma di installazione della piattaforma Web Microsoft (Web PI).

Esistono tre pacchetti popolari per Windows che installano Apache, PHP, MySQL, phpMyAdmin e molti altri strumenti sul tuo computer in

un'unica operazione: XAMPP, WampServer e EasyPHP. Il processo di installazione richiede normalmente solo pochi minuti e, una volta installato il pacchetto, potrebbe essere necessario modificare alcune impostazioni, come vedremo in seguito.

MacOS X

Sia il web server Apache sia PHP sono preinstallati su Mac OS X, ma non sono abilitati per impostazione predefinita. Al posto di utilizzare le versioni preinstallate, ti consiglio di utilizzare **MAMP** , che installa Apache, PHP, MySQL, phpMyAdmin e molti altri strumenti in una sola operazione. Per evitare conflitti con le versioni preinstallate di Apache e PHP, MAMP individua tutte le applicazioni in una cartella dedicata sul disco rigido semplificando la disinstallazione di tutto semplicemente trascinando la cartella MAMP nel Cestino, qualora necessario.

Prima di iniziare, assicurati di aver effettuato l'accesso al tuo computer con privilegi di amministratore.

- Scarica l'immagine gratuita di MAMP;
- Al termine del download, avvia l'immagine del disco e prosegui accettando il contratto di licenza. Prosegui per continuare con il montaggio dell'immagine del disco;
- Segui le istruzioni visualizzate;
- Verifica che MAMP sia stato installato nella cartella Applicazioni.

MAMP di default utilizza porte non standard per Apache e MySQL. A meno che non si utilizzino più installazioni di Apache e MySQL, è necessario modificare le impostazioni della porta come segue:

1. Fare doppio clic sull'icona MAMP in Applicazioni / MAMP. Il tuo browser predefinito dovrebbe avviarsi e presentarti la pagina di benvenuto di MAMP. Nota che l'URL nella barra degli indirizzi del browser inizia con

localhost:8888. Questo indica che il sistema gira in locale e sulla porta 8888 ovvero indica che Apache è in attesa di richieste sulla porta non standard 8888;

2. Apri il pannello di controllo MAMP, che dovrebbe essere in esecuzione sul desktop. Le piccole luci verdi vicino ad Apache Server e MySQL Server indicano che entrambi i server sono in esecuzione;
3. Fai clic sull'icona *Preferenze* e selezionare il tab *Porte* nella parte superiore del pannello. Qui vedrai le porte su cui Apache e MySQL sono in esecuzione ovvero 8888 e 8889;
4. Fai clic su "Imposta le porte Web e MySQL su 80 e 3306" per impostare la configurazione standard;
5. Fai clic su OK e inserisci la password del Mac quando richiesto in modo che MAMP possa riavviare entrambi i server;
6. Quando entrambe le spie sono di nuovo verdi, fare clic su "Apri pagina iniziale" nel pannello di controllo di MAMP. Questo ricarica la pagina di benvenuto di MAMP nel tuo browser. Questa volta, l'URL non dovrebbe avere due punti seguiti da un numero dopo localhost perché Apache ora è in ascolto sulla porta predefinita.

Dove sono i file?

È necessario creare i file in una posizione in cui il server Web possa elaborarli. Normalmente, ciò significa che i file devono trovarsi ad un livello alto del server o in una sottocartella di quel livello. Il percorso predefinito per le impostazioni più comuni è il seguente:

- XAMPP: C:\xampp\htdocs
- WampServer: C:\wamp\www
- EasyPHP: C:\EasyPHP\www
- IIS: C:\inetpub\wwwroot
- MAMP: Macintosh HD:Applications:MAMP:htdocs

Per visualizzare una pagina PHP, è necessario caricarla in un browser utilizzando un URL. L'URL per la pagina principale del server Web nell'ambiente locale è http://localhost/ .

Verificare l'installazione

Dopo aver installato PHP, è una buona idea controllare le sue impostazioni di configurazione. Oltre alle funzionalità principali, PHP ha un gran numero di estensioni opzionali. Entrambi i pacchetti all-in-one e Microsoft Web PI installano tutte le estensioni necessarie per questo ebook. Tuttavia, alcune delle impostazioni di configurazione di base potrebbero essere leggermente diverse. Per evitare problemi imprevisti, controlla che la tua configurazione di PHP sia uguale alla seguente in modo che corrisponda alle impostazioni consigliate.

PHP ha un comando integrato, phpinfo(), che mostra i dettagli di come PHP è configurato sul server. La quantità di dettagli prodotti da questo comando può sembrare un enorme sovraccarico di informazioni ma è preziosa per determinare perché qualcosa funziona perfettamente sul tuo computer locale ma non sul tuo sito web. Di solito il problema si trova nel server remoto che ha

disabilitato una funzione o non ha installato un'estensione opzionale. I pacchetti all-in-one semplificano l'esecuzione di phpinfo():

• XAMPP: fai clic sul collegamento phpinfo nel menu a sinistra della schermata di benvenuto di XAMPP.

• MAMP: fai clic su phpinfo nel menu principale nella parte superiore della pagina iniziale di MAMP.

• WampServer: apri il menu WampServer e fai clic su Localhost. Il collegamento per phpinfo() si trova in Strumenti.

La pagina che troverai sarà qualcosa di simile a questa:

PHP Version 7.3.18

Build Date	May 12 2020 10:48:27
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\xampp2020\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,TS,VC15
PHP Extension Build	API20180731,TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, zip, compress.zlib, compress.bzip2, https, ftps, phar

Dovresti eseguire phpinfo() anche sul tuo server remoto per verificare quali funzionalità sono abilitate. Se le estensioni elencate non sono supportate, parte del codice in questo ebook non funzionerà quando caricherai i tuoi file sul tuo sito web. Qualora dovessi trovare una configurazione diversa sarà necessario modificare il file php.ini. Si tratta di un file molto lungo, che tende a turbare i nuovi arrivati nella programmazione, ma non c'è nulla di cui preoccuparsi. È un semplice testo e uno dei motivi della sua lunghezza è che contiene molti commenti che spiegano le varie opzioni. Detto questo, è una buona idea fare una copia di backup prima di modificare php.ini in caso di errore. Dopo averlo aperto

troverai delle righe che iniziano con un punto e virgola (;) che sono commenti. Le righe che devi modificare non iniziano con un punto e virgola. La maggior parte delle direttive sono precedute da uno o più esempi di come dovrebbero essere impostate. Assicurati di non modificare uno degli esempi commentati per errore. Per le direttive che usano On o Off, basta cambiare il valore in quello raccomandato. Ad esempio, se è necessario attivare la visualizzazione dei messaggi di errore, modificare questa riga:

display_errors = Off

cambiandolo in questo:

display_errors = On

Per impostare il livello di segnalazione degli errori, è necessario utilizzare le costanti PHP, scritte in maiuscolo e con distinzione tra maiuscole e minuscole. La direttiva dovrebbe apparire così:

error_reporting = E_ALL

Dopo aver modificato php.ini, salva il file e riavvia Apache o IIS in modo che le modifiche abbiano effetto. Se il server Web non si avvia, controlla il file di registro degli errori del server. Si può trovare nelle seguenti posizioni:

- XAMPP: nel pannello di controllo di XAMPP, fai clic sul pulsante Registri accanto ad Apache, quindi seleziona Apache error.log.

- MAMP: in Applicazioni: MAMP: registri, fare doppio clic su apache_error.log per aprirlo in Console.
- WampServer: nel menu WampServer, selezionare Apache poi Registro errori Apache.
- EasyPHP: fare clic con il pulsante destro del mouse sull'icona EasyPHP nella barra delle applicazioni e selezionare File di registro poi Apache;
- IIS: il percorso predefinito dei file di registro è C:\inetpub\logs.

La voce più recente nel registro degli errori dovrebbe fornire un'indicazione di ciò che ha impedito il riavvio del server. Utilizza tali informazioni per correggere le modifiche apportate al file php.ini. Se il problema persiste, usa il file di backup che hai salvato prima delle modifiche. Ricomincia e controlla attentamente le modifiche che apporti al file.

Capitolo 4
Codice PHP

A prima vista, il codice PHP può sembrare abbastanza complesso, ma una volta comprese le basi, scoprirai che la struttura è straordinariamente semplice. Se hai già lavorato con qualsiasi altro linguaggio di programmazione, come JavaScript o jQuery, scoprirai che hanno molto in comune.

Ogni pagina PHP deve avere quanto segue:

- L'estensione del file corretta, in genere *.php*
- Tag di apertura e chiusura PHP che circondano ogni blocco di codice PHP

Una tipica pagina PHP utilizzerà alcuni o tutti i seguenti elementi:

- Variabili che fungono da segnaposto per valori sconosciuti o che cambiano
- Matrici per contenere più valori
- Dichiarazioni condizionali per prendere decisioni
- Loop per eseguire attività in modo ripetuto
- Funzioni o oggetti per eseguire attività preimpostate

Diamo una rapida occhiata a ciascuno di questi elementi, iniziando dal nome del file e dai tag di apertura e chiusura.

PHP è un linguaggio usato lato server cioè il server Web elabora il tuo codice PHP e invia solo i risultati, in genere sottoforma di HTML, al browser. Poiché tutte le azioni sono sul server, devi indicare che le tue pagine contengono codice PHP. Ciò comporta due semplici passaggi, vale a dire:

- Assegnare a ogni pagina un'estensione del nome file PHP; il valore predefinito è *.php* ;
- Racchiudere tutto il codice PHP all'interno dei tag PHP.

Il tag di apertura è <?php e il tag di chiusura è ?>. Se inserisci i tag sulla stessa riga del codice circostante, non è necessario che ci sia uno spazio prima del tag di apertura o dopo quello di chiusura, ma deve esserci uno spazio dopo il php nel tag di apertura in questo modo:

```
<p> Questo è HTML con PHP incorporato <? php // codice PHP ?
> . </p>
```

Quando inserisci più di una riga di PHP, è una buona idea mettere il tag di apertura e di chiusura su righe separate per motivi di chiarezza.

```
<?php
// riga di codice PHP
// un'altra riga di codice PHP
?>
```

Potresti imbatterti in <? come versione breve alternativa del tag di apertura, tuttavia, questo tag non funziona su tutti i server. Usa <? php in modo da non avere alcun problema di compatibilità tra server.

PHP è un linguaggio che viene incorporato in pagine Web, ciò significa che è possibile inserire blocchi di codice PHP all'interno di normali pagine Web. Quando qualcuno visita il tuo sito e richiede una pagina PHP, il server la invia al motore PHP, che legge la pagina dall'alto verso il basso alla ricerca di tag PHP. L'HTML non viene alterato ma, ogni volta che il motore PHP incontra un tag <?php, inizia l'elaborazione del codice e continua fino a raggiungere il tag di chiusura ?>. Se il codice PHP produce qualcosa in output, verrà inserito in quel punto della pagina.

Variabili

PHP fa uso anche di variabili ovvero un nome che dai a qualcosa che può cambiare o che non conosci in anticipo. Le variabili in php iniziano sempre con $ (un segno di dollaro). Sebbene il concetto di variabile sembri astratto, usiamo continuamente le variabili nella vita di tutti i giorni. Quando incontri qualcuno per la prima volta, una delle prime cose che chiedi è "Come ti chiami?" Non importa se la persona che hai appena incontrato è Antonio, Filippo o Gerry, la parola "nome" rimane costante. Allo stesso modo, con il tuo conto bancario, i soldi entrano ed escono tutto il tempo ma la quantità disponibile è sempre indicata come saldo. Quindi, "nome" e "saldo" sono variabili quotidiane. Metti un segno di dollaro davanti a loro e hai due variabili PHP già pronte, in questo modo:

$nome

$saldo

Puoi scegliere qualsiasi cosa ti piaccia come nome una variabile, purché tieni a mente le seguenti regole:

- Le variabili iniziano sempre con un simbolo di dollaro ($);
- Il primo carattere dopo il simbolo del dollaro non può essere un numero;
- Non sono ammessi spazi o segni di punteggiatura, ad eccezione del carattere di sottolineatura (_).
- I nomi delle variabili fanno distinzione tra maiuscole e minuscole: $saldoiniziale e $saldoIniziale non sono uguali.

Quando si scelgono i nomi per le variabili, ha senso scegliere qualcosa che ti dica a cosa serve. Le variabili che hai visto finora sono dei buoni esempi. Poiché non puoi utilizzare spazi nei nomi delle variabili, è una buona idea scrivere in maiuscolo la prima lettera della seconda o delle parole successive parole quando le

combini (detta notazione camelCase o a cammello). In alternativa, puoi usare un trattino basso ($saldo_iniziale) detta notazione a serpente o snake_case.

Non cercare di risparmiare tempo utilizzando variabili molto brevi infatti l'uso di $s, $p, $n e $b rende il codice più difficile da capire e ciò rende difficile la scrittura. Ancora più importante, rende gli errori più difficili da individuare. Come sempre, ci sono eccezioni ad una regola. Per convenzione, $i, $j e $k vengono spesso utilizzati per tenere conto del numero di volte in cui è stato eseguito un ciclo e $ e viene utilizzato nel controllo degli errori.

Le variabili ottengono i loro valori da diverse fonti, tra cui:

- Input dell'utente tramite moduli online
- Un database
- Una fonte esterna, come un feed di notizie o un file XML
- Il risultato di un calcolo
- Inclusione diretta nel codice PHP

Ovunque provenga il valore, viene sempre assegnato con un segno di uguale (=), in questo modo:

$variabile = valore;

La variabile va a sinistra del segno di uguale e il valore va a destra e poiché assegna un valore, il segno di uguale viene chiamato operatore di assegnazione.

PHP è scritto come una serie di comandi o istruzioni. Ogni istruzione normalmente dice al motore PHP di eseguire una determinata azione, e deve sempre essere seguita da un punto e virgola, come questo:

```
<?php
Fai questo ;
ora fai questo ;
?>
```

Come per tutte le regole, esiste un'eccezione: puoi omettere il punto e virgola se nel blocco di codice è presente una sola istruzione. Tuttavia, ti consiglio di non farlo perché a differenza di JavaScript, PHP non suppone automaticamente che ci dovrebbe essere un punto e virgola alla fine di una riga. Questo ha un piacevole effetto collaterale: puoi distribuire lunghe istruzioni su più righe e disporre il tuo codice per facilitarne la lettura. PHP, come HTML, ignora gli spazi bianchi nel codice. Al contrario, si basa sui punti e virgola per capire dove termina un comando e inizia quello successivo.

Gli array

In comune con altri linguaggi di calcolo, PHP consente di memorizzare più valori in un tipo speciale di variabile chiamato array. Immagina una matrice come una lista della spesa e sebbene ogni articolo possa essere diverso, puoi fare riferimento a essi collettivamente con un singolo nome. Immagina una variabile $listaSpesa che si riferisce collettivamente a cinque elementi: vino, pesce, pane, uva e formaggio.

I singoli elementi, o elementi dell'array, vengono identificati mediante un numero tra parentesi quadre che segue immediatamente il nome della variabile. PHP assegna il numero automaticamente, ma è importante notare che la numerazione inizia sempre da 0. Quindi il primo elemento dell'array, vino nel nostro esempio, viene indicato come $listaSpesa[0], e non come $listaSpesa[1] come ci si potrebbe aspettare. Per lo stesso motivo, sebbene ci siano cinque elementi, l'ultimo (formaggio) si trova con $listaSpesa[4]. Il numero viene indicato come chiave o indice dell'array e questo tipo di array viene chiamato **array indicizzato** .

PHP utilizza un altro tipo di array in cui la chiave è una parola (o qualsiasi combinazione di lettere e numeri). Ad esempio, un array potrebbe apparire così:

```
$articolo['titolo']='Spugna per cucina';

$articolo['produttore']='Azienda Pippo';

$articolo['quantita'] = '100';

$articolo['id'] = '9238-1-442-036-2';
```

Questo tipo di array è chiamato **array associativo** . Nota che la chiave dell'array è racchiusa tra virgolette (possono essere singole o doppie, non importa) e non deve contenere spazi o punteggiatura.

Le matrici sono una parte importante e utile di PHP infatti sono molto usate. Le matrici vengono ampiamente utilizzate con i database

quando si recuperano i risultati di una ricerca che vengono storicizzati in una serie di matrici.

PHP ha diversi array integrati che vengono automaticamente compilati con informazioni utili. Sono chiamati **array superglobali** e iniziano tutti con un segno di dollaro seguito da un trattino basso. Due che vedrai frequentemente sono $_POST e $_GET. Contengono informazioni passate dai form attraverso in post tramite il protocollo HTTP (Hypertext Transfer Protocol). Gli array superglobali sono tutti array associativi e le chiavi di $_POST e $_GET vengono automaticamente derivate dai nomi degli elementi del modulo o delle variabili in una stringa di query alla fine di un URL. Supponiamo che tu abbia un campo di input di testo chiamato "indirizzo" in un modulo; PHP crea automaticamente un elemento array chiamato $_POST['indirizzo'] quando il modulo è inviato dal metodo post o $_GET['indirizzo'] se usi il metodo GET. $_POST['indirizzo'] contiene il valore immesso da un visitatore nel campo di testo, che consente di visualizzarlo sullo schermo, inserirlo in un database, inviarlo alla posta in arrivo o fare ciò che si desidera.

Valori speciali

Sebbene il testo debba essere racchiuso tra virgolette, esistono tre valori speciali - true, false e null – che non dovrebbero mai essere racchiusi tra virgolette a meno che non si desideri trattarli come testo (o stringhe). I primi due significano cosa ti aspetteresti; l'ultimo, null, significa "niente" o "nessun valore". Questi valori sono molto utili per PHP per prendere delle decisioni, ad esempio se il valore di una variabile è true esegui un comando.

Dai un'occhiata al seguente codice:

```
$variabile = false;
```

Fa esattamente quello che ti aspetti: rende $variabile falso. Ora guarda questo:

```
$variabile = 'false';
```

Questo fa esattamente il contrario di quello che potresti aspettarti: rende $variabile vero! Perché? Perché le virgolette attorno a false lo trasformano in una stringa e PHP considera le stringhe come vere. L'altra cosa da notare su questi valori è che non fanno distinzione tra maiuscole e minuscole. I seguenti esempi sono tutti validi:

```
$variabile = TRUE;

$variabile = tRuE;

$variabile = true;
```

Istruzioni condizionali

La vita è piena di decisioni così come PHP. Se hai esperienza con altri linguaggi di programmazione conosci già buona parte di questo sottocapitolo. Il processo decisionale in PHP utilizza dichiarazioni condizionali ed il più comune usa il se, avvicinandosi molto al linguaggio comune. Nella vita reale, potresti trovarti di fronte alla seguente decisione: se fa caldo, vado in spiaggia.

Nello pseudo-codice PHP, la stessa decisione è simile alla seguente:

```
if (fa caldo) {

vado in spiaggia;

}
```

La condizione da testare è racchiusa tra parentesi tonde mentre l'azione è racchiusa tra parentesi graffe. Questo è il modello decisionale di base:

```
if (la condizione è vera) {

// codice da eseguire se la condizione è vera

}
```

Il codice all'interno delle parentesi graffe viene eseguito solo se la condizione è vera. Se è falso, PHP ignora tutto ciò che è tra parentesi graffe e passa alla sezione successiva del codice. Nel codice precedente abbiamo inserito anche un commento, inizia con i simboli //.

A volte, l'istruzione if è tutto ciò di cui hai bisogno, ma spesso vuoi che venga invocata un'azione se la condizione non è soddisfatta. Per fare questo, usa else, in questo modo:

```
if (la condizione è vera) {

// codice da eseguire se la condizione è vera
```

```
} else {

// codice da eseguire se la condizione è falsa

}
```

Se vuoi più alternative, puoi aggiungere più istruzioni condizionali come questa:

```
if (la condizione è vera) {

// codice da eseguire se la prima condizione è vera

} elseif (la seconda condizione è vera) {

// codice da eseguire se la prima condizione fallisce ma la seconda condizione è vera

} else {

// codice se entrambe le condizioni sono false

}
```

Le dichiarazioni condizionali sono utili solo per verificare che la condizione testata è vera infatti se non è vera, deve essere falsa. Le condizioni dipendono spesso dal confronto di due valori. Questo valore è maggiore di quello? Sono uguali? E così via.

Per verificare l'uguaglianza, PHP utilizza due segni di uguale (==), in questo modo:

```
if ($profilo == 'amministratore') {

// invia alla pagina di amministrazione

} else {

// invia alla pagina utente

}
```

Allo stesso modo puoi usare i simboli maggiore di (>), minore di (<), maggiore o uguale (>=) o minore uguale di (=<), diverso da (!=) per comparare dei valori.

Spesso, confrontare due valori non è sufficiente. PHP consente di impostare una serie di condizioni utilizzando operatori logici per specificare se tutti o solo alcuni devono essere soddisfatti.

Gli operatori logici più importanti in PHP sono elencati nella seguente tabella:

&&	And	$a && $b	Equivale a vero se entrambi $a e $b sono veri
||	Or	$a || $b	Equivale a vero se $a o $b sono true; altrimenti, falso
!	Not	!$a	Equivale a vero se $a non è true

Mostrare risultati

Ovviamente è possibile visualizzare i risultati nella tua pagina web e per farlo esistono due modi in PHP: usare echo o print. Ci sono alcune sottili differenze tra i due, ma puoi considerarli equivalenti. È possibile utilizzare echo con variabili, numeri e stringhe; mettilo semplicemente davanti a qualunque cosa tu voglia visualizzare, in questo modo:

```
$nome = 'Filippo';

echo $nome; // visualizza Filippo

echo 5; // visualizza 5

echo 'David'; // visualizza David
```

Quando si utilizzano echo e print con una variabile, funzionano solo con variabili che contengono un singolo valore. Non è possibile utilizzarli per visualizzare il contenuto di un array o di un'estrazione da un database. Per fare ciò sono necessari i loop anche detti cicli: usi echo o print all'interno del loop per visualizzare ogni elemento singolarmente. Vedremo qualche esempio di questo tipo nei capitoli successivi.

Potresti vedere degli script che usano le parentesi con echo e print, in questo modo:

```
echo ( 'David'); // visualizza David
```

Le parentesi non fanno alcuna differenza e a meno che non ti piaccia scriverle per il gusto di farlo, non utilizzarle.

Quando si desidera visualizzare il valore di una singola variabile o espressione è possibile utilizzare un tag PHP abbreviato in questo modo:

```
<p> Il mio nome è <? = $nome ; ?> . </p>
```

Questo produce esattamente lo stesso output di questo:

```
<p> Il mio nome è <? php echo $nome ; ?> . </p>
```

Poiché si tratta di una scorciatoia, nessun altro codice può trovarsi nello stesso blocco PHP, ma è particolarmente utile quando si incorporano i risultati del database in una pagina Web. È ovvio che il valore della variabile deve essere impostato in un precedente blocco PHP prima di poter utilizzare questo tipo di visualizzazione.

Capitolo 5
Tipi di dati

PHP è ciò che è noto come un linguaggio **debolmente tipizzato** , in pratica, a differenza di altri linguaggi di programmazione (ad esempio Java o C #), a PHP non importa quale tipo di dati archiviate in una variabile.

Il più delle volte, questo è molto conveniente, anche se devi stare attento con l'input dell'utente. Potresti aspettarti che un utente inserisca un numero in un campo di un form, ma PHP non restituirà un errore se l'utente inserisce una parola. Per questo motivo è fondamentale controllare attentamente l'input dell'utente. Sebbene PHP sia tipizzato in modo debole, utilizza i seguenti otto tipi di dati:

- **Integer** : un numero intero, ad esempio 1, 25, 42 o 2006. I numeri interi non devono contenere virgole o punteggiatura per i separatori. Puoi anche usare numeri esadecimali, che dovrebbero essere preceduti da 0x (ad es. 0xFFFFFF, 0x000000).

  - **Floats** : un numero che contiene un punto decimale, come 9.99, 98.6 o 2.1. PHP non supporta l'uso della virgola come punto decimale, comune in molti paesi europei infatti devi usare un punto. Come per i numeri interi, i numeri in virgola mobile non devono separatori per le migliaia.
  - **String** : una stringa è un testo di qualsiasi lunghezza. Può essere breve con zero caratteri (una stringa vuota) e non ha limiti superiori.

- **Boolean** : questo tipo ha solo due valori: true o false.
- **Array** : un array è una variabile in grado di memorizzare più valori, sebbene possa anche non contenerne affatto (un array vuoto). Gli array possono contenere qualsiasi

tipo di dati, inclusi altri array. Un array di array è chiamato array multidimensionale.

- **Object** : un oggetto è un tipo di dati complesso in grado di memorizzare e manipolare valori.
- **Resource** : quando PHP si connette ad una fonte di dati esterna, come un file o un database, memorizza un riferimento ad esso come risorsa.
- **NULL** : questo è un tipo di dati speciale che indica che una variabile non ha valore.

Un importante effetto collaterale della tipizzazione debole di PHP è che se racchiudi un numero intero o in virgola mobile tra virgolette, PHP lo converte automaticamente da una stringa in un numero, permettendoti di eseguire calcoli senza la necessità di alcuna gestione speciale. Sotto questo aspetto è diverso da JavaScript e può avere conseguenze inattese. Quando PHP vede il segno più (+), presume che tu voglia eseguire l'addizione, e quindi prova a convertire stringhe in numeri interi o in virgola mobile, come nell'esempio seguente:

```
$frutti = '2 mele';

$verdure = '2 carote';

echo $frutti + $verdure; // visualizza 4
```

PHP vede che sia $frutti che $verdure iniziano con un numero, quindi estrae il numero e ignora il resto. Tuttavia, se la stringa non inizia con un numero, PHP la converte in 0, come mostrato in questo esempio:

```
$frutti = '2 mele';

$verdure = 'e 2 carote';

echo $frutti + $verdure; // visualizza 2
```

La tipizzazione debole è una croce e delizia ma rende PHP molto semplice per i principianti. Ricorda che spesso è necessario verificare che una variabile contenga il tipo di dati corretto prima di utilizzarlo.

Capitolo 6
I cicli

While e do…while

Un ciclo è una sezione di codice che viene ripetuta fino a quando non viene soddisfatta una determinata condizione. I loop (o cicli) sono spesso controllati impostando una variabile che conta il numero di iterazioni. Aumentando la variabile di una ogni volta, il loop si ferma quando la variabile raggiunge un numero preimpostato. I loop sono inoltre controllati eseguendo ogni elemento di un array. Quando non ci sono più elementi da elaborare, il ciclo si interrompe. I loop contengono spesso istruzioni condizionali, quindi sebbene siano molto semplici nella struttura, possono essere utilizzati per creare codice che elabora i dati in modi complesso.

Il tipo più semplice di ciclo è chiamato ciclo while. La sua struttura di base si presenta così:

```
while (la condizione è vera) {

fai qualcosa;

}
```

Il codice seguente mostra ogni numero compreso tra 1 e 100 in un browser. Inizia impostando una variabile ($i) ad 1 e quindi utilizza la variabile come contatore per controllare il loop, oltre a visualizzare il numero corrente sullo schermo.

```
$i = 1; // imposta il contatore a 1

while ($i <= 100) {

  echo "$i <br>";

  $i++; // aumenta il contatore di 1
```

```
}
```

Una variante del ciclo while utilizza la parola chiave do e segue questo modello di base:

```
do {
  codice da eseguire
} while (condizione da testare);
```

La differenza tra un do…while e un ciclo while è che il codice all'interno del blocco do viene eseguito almeno una volta, anche se la condizione non è mai vera. Il seguente codice mostra il valore di $i una volta, anche se è maggiore del massimo previsto.

```
$i = 1000;
do {
  echo "$i <br>";
  $i++; // aumenta il contatore di 1
} while ($i <= 100);
```

Il pericolo con il do…while consiste nel dimenticare di impostare una condizione che porta alla fine del loop o l'impostazione di una condizione impossibile. Questo è noto come un ciclo infinito che blocca il computer o provoca l'arresto anomalo del browser.

For e foreach

Il ciclo for è meno incline a generare un ciclo infinito perché è necessario dichiarare tutte le condizioni del ciclo nella prima riga. Il ciclo for utilizza il seguente modello di base:

```
for (inizializzazione; condizione; codice da eseguire dopo ogni iterazione) {

  codice da eseguire

}
```

Il codice seguente fa esattamente lo stesso del ciclo while precedente, mostrando tutti i numeri da 1 a 100:

```
for ($i = 1; $i <= 100; $i ++) {

  echo "$i <br>";

}
```

Le tre espressioni tra parentesi controllano l'azione del ciclo (nota che sono separate da punti e virgola, non da virgole):

1. La prima espressione viene eseguita prima dell'inizio del loop. In questo caso, imposta il valore iniziale della variabile contatore $i su 1.
2. La seconda espressione imposta la condizione che determina per quanto tempo il ciclo dovrebbe continuare a funzionare. Può essere un numero fisso, una variabile o un'espressione che calcola un valore.
3. La terza espressione viene eseguita alla fine di ogni iterazione del ciclo. In questo caso, aumenta di $i di 1, ma non c'è nulla che ti impedisca di incrementare di 4, 10 o 50. Ad esempio, la sostituzione di $i++ con $i+=10 in questo esempio visualizzerà 1, 11, 21, 31 e così via.

L’ultimo tipo di ciclo in PHP viene utilizzato con matrici e oggetti. Assume due forme diverse, entrambe utilizzano variabili temporanee per gestire ciascun elemento. Se devi usare solo il valore dell'elemento, il ciclo foreach assume la forma seguente:

```
foreach (nomeVariabile as elemento) {

  fai qualcosa con l'elemento

}
```

L'esempio seguente crea un ciclo sull'array $listaSpesa e visualizza il nome di ciascun elemento:

```
$listaSpesa = ['vino', 'pesce', 'pane', 'uva', 'formaggio'];

foreach ($ listaSpesa as $item) {

  echo $item. '<br>';

}
```

Sebbene l'esempio precedente utilizzi un array indicizzato, è anche possibile utilizzare la forma semplice del ciclo foreach con un array associativo. La forma alternativa del ciclo foreach consente di accedere sia alla chiave che al valore di ciascun elemento. Prende questa forma leggermente diversa:

```
foreach (nomeVariabile as chiave => valore) {

  fai qualcosa con chiave e valore }
```

Per terminare prematuramente un ciclo quando viene soddisfatta una determinata condizione, inserire la parola chiave break all'interno di un'istruzione condizionale. Non appena lo script viene interrotto, si uscirà dal ciclo.

Per saltare la valutazione del codice in un ciclo quando viene soddisfatta una determinata condizione, utilizzare la parola chiave continue. Invece di uscire dal ciclo, ritorna all'inizio del ciclo e si occupa dell'elemento successivo. Ad esempio, il ciclo seguente salta l'elemento corrente se $foto non ha valore:

```
foreach ($immagini as $foto) {
if (empty ($foto)) continue;
// codice per visualizzare una foto
}
```

Capitolo 7
Funzioni

Le funzioni offrono un modo conveniente per eseguire operazioni eseguite di frequente. Oltre al gran numero di funzioni integrate, PHP ti consente di creare le tue. Il vantaggio principale consiste nello scrivere il codice una sola volta, piuttosto che doverlo scrivere nuovamente ovunque tu ne abbia bisogno. Non solo accelera lo sviluppo, ma semplifica anche la lettura e la manutenzione del codice. Se c'è un problema con il codice nella tua funzione, puoi aggiornarlo in un solo punto anziché cercare ovunque. Inoltre, le funzioni di solito accelerano l'elaborazione delle tue pagine.

Costruire le tue funzioni in PHP è facile. È sufficiente racchiudere un blocco di codice in una coppia di parentesi graffe e utilizzare la parola chiave function per dare un nome alla nuova funzione. Il nome della funzione è sempre seguito da una coppia di parentesi. Il seguente esempio mostra la struttura di base di una funzione personalizzata:

```
function saluta() {

  echo 'Ciao!';

}
```

Puoi invocare questa funzione semplicemente scrivendo saluta(); in un blocco di codice PHP e sullo schermo vedrai Ciao!. Questo tipo di funzione esegue sempre esattamente la stessa operazione. Affinché le funzioni rispondano alle circostanze, è necessario passare loro dei valori come argomenti.

Supponiamo che tu voglia adattare la funzione in modo che mostri il nome di qualcuno. Puoi farlo inserendo una variabile tra parentesi nella dichiarazione di funzione. La stessa variabile viene quindi utilizzata all'interno della funzione per visualizzare qualunque valore venga passato alla funzione. Per passare più di un argomento a una

funzione, separare le variabili con virgole tra parentesi aperte. Ecco come appare la funzione adesso:

```
function saluta($nome) {

  echo "Ciao $nome!";

}
```

Ora puoi usare questa funzione all'interno di una pagina per visualizzare il valore di qualsiasi variabile passata a saluta(). Ad esempio, se hai un modulo online che salva il nome di qualcuno in una variabile chiamata $visitatore e Antonio visita il tuo sito, puoi salutarlo in modo personalizzato invocando la funzione saluta($visitatore); nella tua pagina.

È anche importante capire che le funzioni creano un ambiente separato che è come una scatola nera. Normalmente ciò che accade all'interno della funzione non ha alcun impatto sul resto dello script, a meno che non restituisca un valore. Le variabili all'interno di una funzione rimangono esclusive della funzione. Questo esempio dovrebbe illustrare cosa intendo:

```
function duplica ($numero) {

$numero *= 2;

echo '$numero è '. $numero. '<br>';

}

$numero = 4;

duplica($numero);

echo '$numero è '. $numero;
```

Le prime quattro righe definiscono una funzione chiamata duplica(), che accetta un numero, lo raddoppia e lo visualizza sullo schermo. Il resto dello script assegna il valore 4 a $numero. Quindi passa $numero come argomento a duplica(). La funzione elabora $numero e visualizza 8. Al termine della funzione, $numero viene visualizzato

sullo schermo dalla funzione echo. Questa volta il valore sarà 4 e non 8 perché il valore non è stato restituito dalla funzione ma solo usato per la visualizzazione a schermo.

Ciò dimostra che $numero nello script principale è totalmente estraneo alla variabile con lo stesso nome all'interno della funzione. Questo è noto come **scope** o ambito della variabile. Anche se il valore della variabile cambia all'interno di una funzione, le variabili con lo stesso nome all'esterno non sono interessate. Per evitare confusione, è una buona idea utilizzare nomi di variabili nel resto dello script diversi da quelli utilizzati all'interno delle funzioni. Questo non è sempre possibile, quindi è utile sapere che le funzioni sono delle piccole scatole nere e normalmente non hanno alcun impatto diretto sui valori delle variabili nel resto dello script. Un altro aspetto importante è che una funzione non può normalmente accedere ai valori nello script esterno a meno che non vengano passati alla funzione come argomenti.

Esiste più di un modo per una funzione per modificare il valore di una variabile data in input, ma il metodo più importante è utilizzare la parola chiave return e assegnare il risultato alla stessa variabile o ad un'altra. Questo può essere dimostrato modificando la funzione duplica() in questo modo:

```
function duplica ($numero) {

return $numero *= 2;

}

$num = 4;

$raddoppiato = duplica($num);

echo '$num è: '. $num. '<br>';

echo '$raddoppiato è: '. $raddoppiato;
```

Questa volta, ho usato nomi diversi per le variabili per evitare di confonderle. Ho anche assegnato il risultato di duplica($num) a una nuova variabile. Il vantaggio di fare questo è che sono ora disponibili

sia il valore originale che il risultato del calcolo. Non sempre è necessario mantenere il valore originale, ma a volte può essere molto utile.

Sebbene le funzioni non cambino normalmente il valore delle variabili passate come argomenti, ci sono occasioni in cui si desidera cambiare il valore originale anziché avere un valore di ritorno. Per fare ciò, quando si definisce la funzione, si antepone il parametro che si desidera modificare con una e commerciale (&), in questo modo:

```
function duplica(&$numero) {

$numero *= 2;

}
```

Nota che questa versione della funzione duplica() non stampa a video il valore della variabile, né restituisce il valore del calcolo. Poiché il parametro tra parentesi è preceduto da una e commerciale (&), il valore originale di una variabile passato come argomento alla funzione verrà modificato. Questo è noto come passaggio per riferimento.

Il seguente codice dimostra l'effetto:

```
$num = 4;

echo '$num è: ' . $num . '<br>';

duplica($num);

echo '$num ora è: ' . $num;
```

Se la tua funzione viene trovata nella stessa pagina in cui viene utilizzata, non importa dove la dichiari; può essere prima o dopo essere stata utilizzata. È una buona idea, tuttavia, dichiarare le funzioni insieme, nella parte superiore o inferiore di una pagina. Questo le rende più facili da trovare e manutenere. Le funzioni utilizzate in più di una pagina vengono archiviate in modo ottimale in un file esterno incluso in ciascuna pagina. L'inclusione di file esterni

avviene tramite include e request. Quando le funzioni sono memorizzate in file esterni, è necessario includere il file esterno prima di usare una funzione.

Require e include

La possibilità di includere il contenuto di un file all'interno di un altro è una delle funzionalità più interessanti di PHP ed è anche una delle più facili da implementare. La maggior parte delle pagine di un sito Web condivide elementi comuni, come intestazione, piè di pagina e menu di navigazione. Puoi modificare l'aspetto di questi elementi in tutto il sito modificando le regole di stile in un foglio di stile esterno. Ma i CSS hanno solo una capacità limitata di modificare il contenuto degli elementi della pagina. Se vuoi aggiungere un nuovo elemento al tuo menu, devi modificare il codice HTML per ogni pagina che lo visualizza.

PHP supporta le inclusioni lato server (SSI) ovvero un file esterno che contiene codice dinamico o HTML (o entrambi) che si desidera incorporare in più pagine. PHP unisce il contenuto in ciascuna pagina Web sul server e poiché ogni pagina utilizza lo stesso file esterno, puoi aggiornare un menu o un altro elemento comune modificando e caricando un singolo file con un grande risparmio di tempo.

La possibilità di includere codice da altri file è una parte fondamentale di PHP. Tutto ciò che è necessario è utilizzare uno dei comandi include di PHP ed indicare al server dove trovare il file.

PHP dispone di quattro comandi che possono essere utilizzati per includere il codice da un file esterno:

- include
- include_once
- require
- require_once

Fanno tutti praticamente la stessa cosa, quindi perché averne quattro a disposizione? La differenza fondamentale è che include cerca di continuare l'elaborazione di uno script, anche se manca il

file esterno, mentre require è usato nel senso inverso: se il file manca, il motore PHP interrompe l'elaborazione e genera un errore. In termini pratici, questo significa che dovresti usare include se la tua pagina rimarrebbe utilizzabile anche senza il contenuto del file esterno. Utilizza require se la pagina dipende dal file esterno.

Gli altri due comandi, include_once e require_once, funzionano allo stesso modo, ma impediscono che lo stesso file venga incluso più di una volta in una pagina, ciò è particolarmente importante quando si includono file che definiscono funzioni o classi. Il tentativo di definire una funzione o una classe più di una volta in uno script provoca un errore irreversibile. Pertanto, l'utilizzo di include_once o request_once garantisce che le funzioni e le classi siano definite una sola volta, anche se lo script tenta di includere il file esterno più di una volta.

Per includere un file esterno, usa uno dei quattro comandi seguito dal percorso del file come stringa: in altre parole, il percorso del file deve essere tra virgolette (singole o doppie, non importa). Il percorso del file può essere assoluto o relativo al documento corrente. Ad esempio, funzionerà in uno dei seguenti modi (purché esista il file di destinazione):

```
include 'includes/menu.php';

include 'C:/xampp/htdocs/phpsols/includes/menu.php';

include '/Applications/MAMP/htdocs/phpsols/includes/menu.php';
```

Quando si utilizza un percorso file relativo, si consiglia di utilizzare ./ per indicare che il percorso inizia nella cartella corrente. Pertanto, è più efficiente riscrivere il primo esempio in questo modo:

```
include './includes/menu.php';
```

Ciò che non funziona è l'utilizzo di un percorso di file relativo alla radice del sito, in questo modo:

```
include '/includes/menu.php';
```

Ecco come includere un footer comune a più pagine:

```php
<?php include './includes/footer.php' ; ?>
```

Quando il motore PHP rileva un comando include, interrompe l'elaborazione di PHP all'inizio del file esterno e riprende nuovamente alla fine. Se si desidera che il file esterno utilizzi il codice PHP, il codice deve essere racchiuso tra tag PHP. Poiché il file esterno viene elaborato come parte del file PHP che lo include, un file include può avere qualsiasi estensione.

Alcuni sviluppatori utilizzano .inc come estensione del nome file per chiarire che il file deve essere incluso in un altro file. Tuttavia, la maggior parte dei server considera i file .inc come testo normale. Ciò comporta un rischio per la sicurezza se il file contiene informazioni riservate, come nome utente e password del database. Se il file è archiviato nella cartella principale del tuo sito Web, chiunque scopra il nome del file può semplicemente digitare l'URL nella barra degli indirizzi del browser e il browser visualizzerà tutti i dati sensibili!

D'altra parte, qualsiasi file con estensione .php viene automaticamente inviato al motore PHP per l'analisi prima di essere inviato al browser. Finché le tue informazioni sensibili si trovano all'interno di un blocco di codice PHP e in un file con estensione .php, queste non saranno esposte. Ecco perché alcuni sviluppatori utilizzano .inc.php come include in modo da avere una doppia estensione. La parte .inc ti ricorda che si tratta di un file include, ma i server sono interessati solo al .php alla fine, il che assicura che tutto il codice PHP sia analizzato correttamente.

Per molto tempo, ho seguito la convenzione di usare .inc.php per i file include. Ma dal momento che immagazzino tutti i miei file include in una cartella separata chiamata include, ho deciso che la doppia estensione è superflua infatti ora uso solo .php. Quale convenzione scegliere dipende da te, ma usare .inc è la meno sicura.

Per essere sicuri che una pagina esista puoi verificare tramite delle funzioni che PHP offre nativamente. Se una pagina sarebbe priva di significato senza il file include, è necessario reindirizzare l'utente a una pagina di errore se il file include è mancante.

Un modo per farlo consiste nel generare un'eccezione, in questo modo:

```
<?php
$file = './includes/menu.php' ;
if ( file_exists ( $file ) && is_readable ( $file )) {
  include $file ;
} else {
  throw new Exception ( " $file non trovato" );
}
?>
```

Una caratteristica utile dei file include di PHP è che possono essere posizionati ovunque, purché la pagina con il comando include sappia dove trovarli. I file da includere non è nemmeno necessario che siano all'interno della radice del tuo server web, ciò significa che è possibile proteggere i file che contengono informazioni riservate, come le password, in una directory privata (cartella) a cui non è possibile accedere tramite un browser. Quindi, se la tua società di hosting fornisce un'area di archiviazione esterna alla radice del tuo server, dovresti prendere in considerazione l'opportunità di inserire lì i tuoi file da includere.

Ma perché non è possibile inserire un riferimento alla radice del sito? Quando si fa clic su un collegamento per passare a un'altra pagina, il percorso nel tag <a> indica al browser come passare dalla pagina corrente a quella successiva. La maggior parte degli strumenti di creazione web specifica il percorso relativo al documento corrente. Se la pagina di destinazione si trova nella stessa cartella, viene utilizzato solo il nome file. Se è superiore di un livello rispetto alla pagina corrente, il nome del file è preceduto da ../. Questo è noto come percorso o collegamento relativo al documento.

L'altro tipo di collegamento inizia sempre con una barra, che indica la radice del sito. Il vantaggio di un percorso relativo alla radice del sito è che non importa quanto sia profonda la pagina corrente nella gerarchia del sito, la barra (/) garantisce che il server Web cercherà

dal livello superiore del sito. Sebbene i collegamenti relativi alla radice del sito siano molto più facili da leggere, PHP non riesce a gestirli.

È possibile convertire un percorso relativo alla radice del sito in uno assoluto concatenando la variabile globale $_SERVER ['DOCUMENT_ROOT'] all'inizio del percorso, in questo modo:

```
include $_SERVER['DOCUMENT_ROOT']./includes/nomefile.php';
```

La maggior parte dei server supporta $_SERVER['DOCUMENT_ROOT'], ma è necessario controllare la sezione Variabili PHP nei dettagli di configurazione visualizzati da phpinfo() per essere sicuri che funzioni.

Conclusioni

Sicuramente in questo ebook non abbiamo affrontato tutti i casi d’uso, tutte le possibili combinazioni che potresti trovare mentre programmi in PHP. Abbiamo analizzato le basi, partendo da una solida teoria sulla quale costruire alcuni esempi per facilitare la comprensione. Se hai imparato tutte le tecniche descritte qui, sei sulla buona strada per diventare un bravo sviluppatore PHP e con un po' più di sforzo, entrerai nel livello avanzato.

Non ti preoccupare se hai dovuto faticare a capire, ripassa i capitoli e vedrai che sarà tutto più chiaro. Non dimenticare che più ti alleni, più tutto diventa facile.

Se stai pensando: "Come posso ricordare tutto questo?", la risposta è che non è necessario ricordare tutto. Siamo nell’era del digitale quindi molti prima di te avranno avuto dubbi su come usare un loop ed è facile sbagliare se sei alle prime armi. Non vergognarti di cercare le cose. Ciò che rende semplice la programmazione per il Web non è una conoscenza enciclopedica delle funzioni e delle classi di PHP, ma una solida teoria su come funzionano le istruzioni condizionali, i loop e altre strutture che controllano il flusso di uno script.

Quando sei in grado di analizzare i tuoi progetti in termini di "se ciò accade, cosa dovrebbe succedere dopo?" sei già ad un livello successivo e sei tu che controlli il codice. Consulto frequentemente il manuale online di PHP e per me è come un dizionario, il più delle volte, voglio solo controllare di ricordare bene. Spesso trovo che qualcosa attira la mia attenzione e apre nuovi orizzonti, magari una nuova funzione che consente di risparmiare tempo e qualche riga di codice. Ti ringrazio per aver letto questo ebook e spero che questo sia solo il punto di partenza per costruire dei bellissimi siti Web, con molte funzioni in PHP, sfruttando tutte le sue potenzialità.

MySQL

Premessa

Attualmente sviluppare un’applicazione richiede diverse competenze, che si tratti di un’app per smartphone, un’applicazione Web o semplicemente un sito Web è necessario conoscere qualche linguaggio di programmazione così come diverse nozioni sulle reti e sulla comunicazione Web.

Questi framework e linguaggi si evolvono rapidamente e spesso i programmatori non riescono a stare al passo degli aggiornamenti proposti ma in tutto ciò esiste una certezza. Sicuramente avrai bisogno di memorizzare i tuoi dati in un database quindi è fondamentale fare la scelta giusta in base al progetto che stai sviluppando. Una buona architettura insieme ad idee chiare ti aiuteranno sicuramente a raggiungere un ottimo risultato.

MySQL può sicuramente esserti d’aiuto perché con la sua semplicità e con le sue performance puoi creare delle applicazioni o siti Web con tempi di risposta davvero bassi soprattutto se sei in grado di sfruttare al meglio tutte le possibilità offerte da questo database. Nel corso di questo ebook affronteremo diversi aspetti: dai punti di forza di MySQL alla sua struttura, dalle tabelle alle relazioni tra tabelle.

Capitolo 1
Cos’è un DB

Un database (anche detto DB) consente la memorizzazione di dati in maniera persistente infatti ogni giorno una grande quantità di dati vengono salvate nei database. Pensa ai database dei grandi e-commerce come Amazon, pensa a quanto sono grandi ed efficienti i database di Facebook per poter memorizzare i post, le foto ed i video di ognuno di noi.

Inizialmente bisogna fare una distinzione tra database e DBMS infatti dobbiamo capire bene di cosa stiamo parlando. Un DBMS è un programma (software) che gestisce uno o più database pertanto ha il compito di gestire gli utenti ed i loro permessi, gestisce la sicurezza e l’ottimizzazione dei database contenuti ecc. Un database, invece, si occupa di memorizzare i dati in modo tale che sia semplice recuperarli, rispettando tutti i vincoli imposti. Di quali vincoli stiamo parlando? Per vincoli si intendono delle regole da rispettare, ad esempio, assicurarsi che la quantità di un articolo memorizzato sia un valore numerico o che l’identificativo di un articolo sia di una determinata lunghezza ed univoco.

Ad oggi esistono diversi tipi di DBMS ma sostanzialmente possono essere rinchiusi in due categorie: **NoSQL** e **RDBMS** . I database NoSQL sono nati di recente e sono molto utilizzati quando si hanno moltissimi dati, pensa ad esempio, agli utenti di Facebook o agli ordini di Amazon. Questo tipo di database è particolarmente incentrato sulla scalabilità infatti sono in grado di adattarsi facilmente all’aumentare della potenza di calcolo. Immagina cosa succede su Amazon durante il Black Friday, si contano molti più accessi, molti più ordini e molti più utenti in poco tempo. I database NoSQL riescono ad adattarsi facilmente garantendo elevate prestazioni e grande flessibilità perché non sono strettamente legati alle **relazioni** tra i dati.

I database di tipo RDBMS sono basati proprio su relazioni (è la R dell'acronimo) tra i dati infatti ogni dato memorizzato segue una struttura rigida. La struttura consente di impostare delle regole e dei vincoli, ad esempio, possiamo garantire ad ogni inserimento che il campo data_di_nascita sia effettivamente una data e non una stringa. In questo modo sarà più facile, successivamente, eseguire un ordinamento o filtrare i risultati. Le relazioni si basano su righe e colonne quindi i dati sono organizzati in tabelle con degli indici ed eventualmente chiavi primarie per definire dei valori univoci nella tabella.

Un'altra importante differenza tra questi tipi di database, tuttavia, consiste nel garantire le **proprietà ACID** . I database NoSQL non le garantiscono pertanto presta molta attenzione nel caso in cui tu sia interessato ad usarli e valuta bene per tutti i tuoi casi d'uso.

Le proprietà ACID si riferiscono a:

- Atomicità ovvero garantire che una transazione venga eseguita certamente o non eseguita affatto
- Coerenza cioè i dati saranno sicuramente coerenti tra loro e quindi conformi allo schema del database
- Isolamento indica che ogni transazione è separata dalle altre
- Durabilità consente di ripristinare i dati all'ultimo stato conosciuto

È probabile, quindi, che se stai pensando di usare un database NoSQL per un software di fatturazione probabilmente non stai facendo la scelta giusta. Un database NoSQL può esserti d'aiuto nel caso in cui tu voglia creare dei grafici, creare delle applicazioni real-time ad esempio legate alla borsa ecc.

Se i database NoSQL sono nati verso gli anni 2000, i database relazionali sono molto più datati, infatti, a partire dal 1970 sono stati ampiamente utilizzati continuando ad evolversi per garantire migliori prestazioni e sempre più funzionalità. Questa evoluzione, oggi, si esprime in un prodotto molto facile da usare tanto che potresti

associarlo ad un comune foglio Excel. Creare una tabella, popolarla ed esportarla risulta davvero semplice così come lo è collegare un database ad un'applicazione sviluppata in Java, PHP, Perl o altri linguaggi.

Una tabella crea delle relazioni al suo interno, ad esempio, potremmo creare una tabella con tutti gli smartphone Samsung usando come colonne: nome commerciale, nome del modello, anno di produzione, processore ecc. Immaginiamo, però, che due o più smartphone montino lo stesso processore, questo sarebbe un **dato rindondante** cioè ripetuto su due o più righe. A questo scopo si possono creare delle relazioni tra tabelle in modo da evitare questa ridondanza e mantenere le tabelle più compatte e con dati altamente coesi tra loro. Esistono vari tipi di relazioni possibili che approfondiremo nei prossimi capitoli.

Probabilmente hai già sentito parlare di SQL tanto che è spesso presente in molti nomi di database (MySQL, PostgreSQL, Sqlite ecc). SQL è l'acronimo di *Structured Query Language* ed indica il linguaggio che i database relazionali possono capire ed interpretare per creare database, memorizzare i dati, recuperare i dati, modificarli e tanto altro. Bisogna ricordare che SQL consente anche di gestire ed amministrare i database, consentendo l'accesso a diversi utenti.

Le classiche operazioni che vengono effettuate su un database sono creazione, lettura, modifica e cancellazione di dati. Queste operazioni sono dette **operazioni CRUD** che è l'acronimo di *Create, Read, Update e Delete* .

Capitolo 2
Punti di forza

Proseguendo sulla nostra analisi dei database relazionali ed essendo certi che si adattano meglio al nostro scopo, cerchiamo di capire perché usare MySQL piuttosto che un altro database relazionale. Innanzitutto, MySQL è nato nel lontano 1996 quindi stiamo parlando di un prodotto molto maturo che si è evoluto nel corso del tempo. Si è evoluto soprattutto grazie alla comunità perché si tratta di un progetto open source e libero quindi chiunque può analizzare il codice sorgente, modificarlo o contribuire per migliorarlo.

I punti di forza di MySQL sono tanti e il più importante riguarda la sua popolarità. Usare un software molto diffuso ti consente di individuare soluzioni ad ogni problema perché probabilmente qualcuno si è posto lo stesso problema prima di te. Nella maggior parte dei casi, una ricerca su Google eviterà di farti perdere molto tempo.

Un altro punto di forza è certamente la sua affidabilità infatti garantisce il suo funzionamento 24 ore al giorno, 7 giorni su 7 ed è anche per questo che viene spesso usato per i siti Web insieme a CMS come Wordpress.

MySQL può essere installato su qualsiasi piattaforma infatti funziona bene sia su Windows, sia su Linux sia su macOS.

È molto apprezzato, oltre a tutto quello che abbiamo già detto, per come svolge il suo lavoro. Questo è il punto fondamentale perché è contraddistinto da un’alta efficienza sia con pochi che con moltissimi dati. È possibile ottimizzare il database a proprio piacimento, ottimizzando una query per estrarre più velocemente i dati, inserendo degli indici nelle tabelle per renderle più accessibili nella ricerca e tanto altro.

Infine, ma non meno importante, è davvero semplice integrarlo con un'applicazione nuova o preesistente scritta in un linguaggio di programmazione come *Java, Python, C, C++* o addirittura *Node.js* .

Non è tutto oro quel che luccica infatti MySQL non è esente da problemi. Nonostante sia stato progettato davvero bene inizia ad avere qualche acciacco soprattutto se paragonato a nuovi prodotti basati proprio su MySQL. Esistono infatti dei *fork* (progetti basati su MySQL ma sviluppati da terzi) che hanno prestazioni leggermente migliori, si parla del 5% circa.

L'aspetto positivo, tuttavia, riguarda la facilità con cui è possibile migrare ad uno di questi fork, molte volte è sufficiente esportare il database e rieseguire le query sul nuovo per completare la migrazione con successo.

Capitolo 3
Installazione

Esistono diverse versioni di MySQL e in questo ebook preferiamo usare la *Community Edition* in quanto gratuita. Questa versione è sottoposta a licenza GPL perciò può essere scaricata in modo gratuito e mette a disposizione diverse funzionalità che possono tornare utili.

Linux

L'installazione su piattaforme basate su Linux è davvero semplice infatti si può scaricare sottoforma di archivio *.tar.gz* o come pacchetto grazie ad *apt* o *yum* .

Prendiamo in considerazione il package manager *apt,* basterà digitare i seguenti comandi:

sudo apt update

sudo apt install mysql-server

Questo comando installerà MySQL, ma non ti chiederà di impostare una password o apportare altre modifiche alla configurazione.

Per le nuove installazioni, ti consigliamo di eseguire lo script di sicurezza. Questo comando modifica alcune delle opzioni predefinite che sono meno sicure per gli accessi root remoti e gli utenti di esempio. Nelle versioni precedenti di MySQL, era possibile inizializzare manualmente anche la directory dei dati ma questo passaggio viene eseguito automaticamente.

Digita il seguente comando nel terminale:

sudo mysql_secure_installation

Questo comando ti guiderà attraverso una serie di istruzioni in cui puoi apportare alcune modifiche alle opzioni di sicurezza della tua installazione MySQL. Innanzitutto, ti verrà chiesto se desideri configurare il plug-in *Validate Password* , che può essere utilizzato per testare la validità della tua password MySQL. Indipendentemente dalla tua scelta, il prossimo step sarà quello di impostare una password per l'utente *root* di MySQL. Immetti la password e conferma la tua scelta. Da adesso, è possibile premere il tasto Y e quindi INVIO per accettare le impostazioni predefinite per tutte le domande successive. Ciò rimuoverà alcuni utenti anonimi e il database di test, disabiliterà gli accessi root remoti e caricherà queste nuove regole in modo che MySQL rispetti le modifiche apportate.

Se l’installazione è andata a buon fine e non vedi alcun messaggio di errore puoi eseguire questo comando nel tuo terminale per l’autenticazione:

mysql -u root -p

Dopo aver digitato questo comando puoi creare un nuovo utente ed assegnargli una password sicura:

mysql> CREATE USER ‘pippo’@’localhost’ IDENTIFIED BY ‘password’;

mysql> GRANT ALL PRIVILEGES ON *.* TO ‘pippo’@’localhost’ WITH GRANT OPTION;

mysql> exit

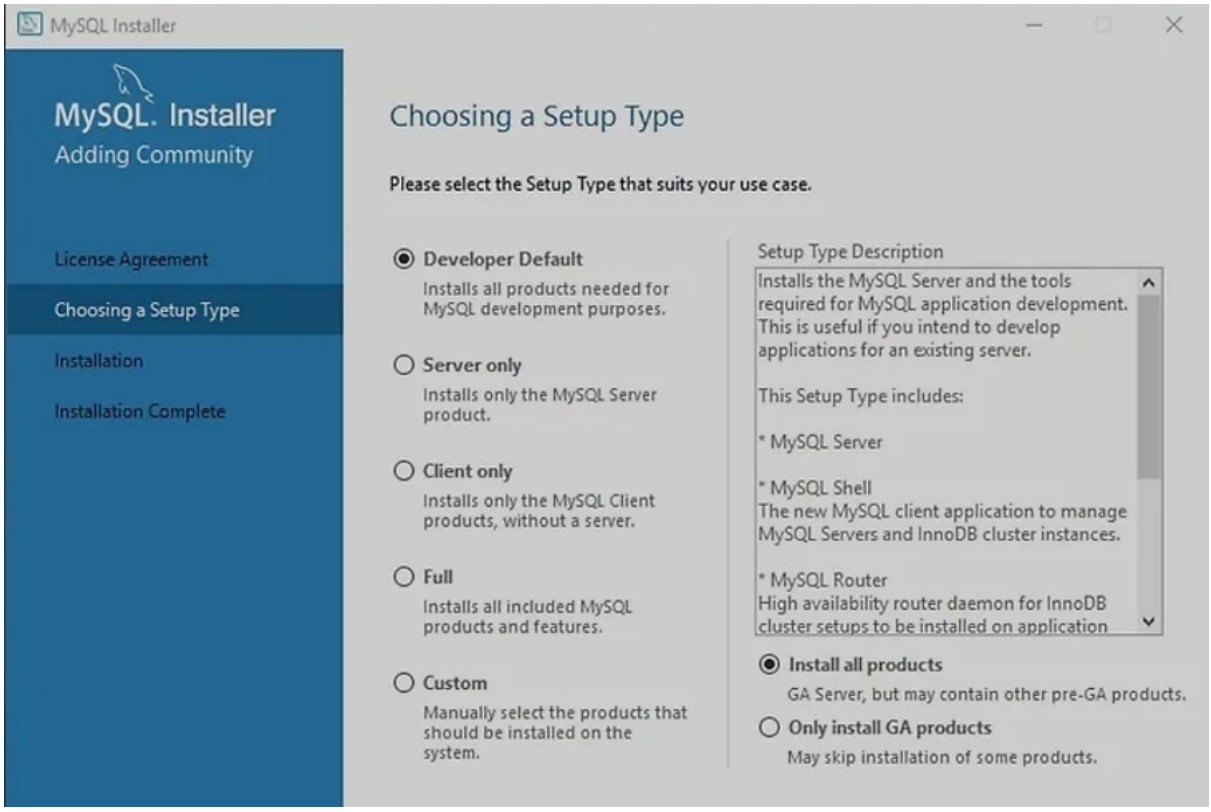
Il primo comando consente di creare l’utente pippo con la password scelta, il secondo concede al nuovo utente i privilegi appropriati. Nell’esempio sono stati concessi i privilegi a tutte le tabelle all'interno del database, nonché il potere di aggiungere, modificare e rimuovere i privilegi degli utenti. L’ultimo comando serve per uscire da MySQL.

Windows

In Windows il processo di installazione è completamente guidato da un'interfaccia grafica che spiega in maniera completa ed esaustiva quali sono gli step necessari per l'installazione e cosa si sta installando.

Per prima cosa è necessario collegarsi al sito ufficiale https://dev.mysql.com/downloads/windows/installer/8.0.html per ottenere la Community Edition. Dopo aver cliccato su "Download" del pacchetto più piccolo in termini di dimensioni possiamo lanciare l'eseguibile scaricato. È anche possibile scaricare il pacchetto di dimensioni più grandi, risparmiando un po' di tempo in fase di installazione. Nella prima schermata viene chiesto cosa vogliamo installare: solo il server, solo il client, un'installazione completa ecc.

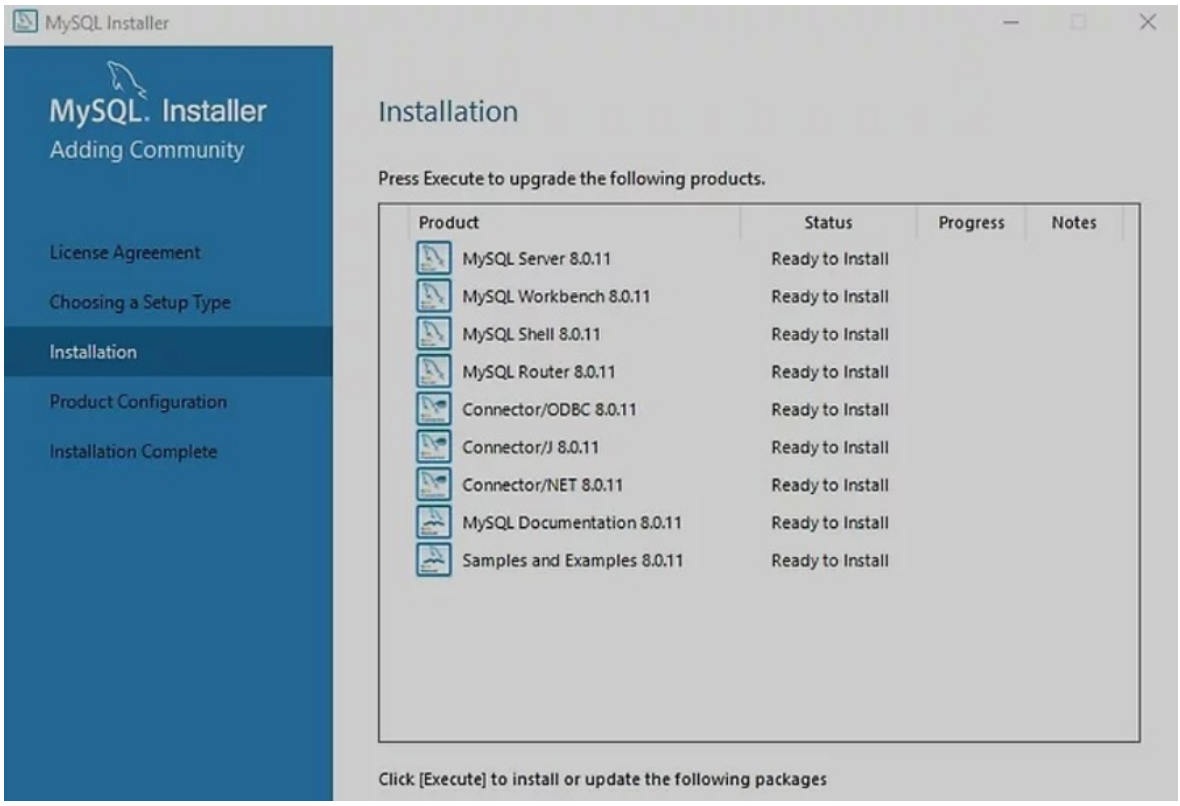
Selezioniamo la prima opzione (Developer Default) e installeremo tutti i prodotti che vediamo nella colonna di destra. Proseguiamo come nell'immagine che segue:



Successivamente ci viene richiesto che tipo di architettura vogliamo usare pertanto selezioneremo *Standalone MySQL Server / Classic MySQL Replication* . Se vogliamo installare il database sul PC locale per motivi di sviluppo o per prendere confidenza con MySQL scegliamo *Development Machine* , se stiamo configurando un server dedicato scegliamo *Server Machine* altrimenti *Dedicated Machine* se si tratta di un server su cui sarà ospitato solo MySQL. Seleziona il protocollo TCP/IP e assicurati che sia impostata la porta di default per MySQL ovvero la 3306.

Alla fine di queste configurazioni può iniziare l'installazione del software che richiederà diversi minuti a seconda dell'hardware su cui state installando.

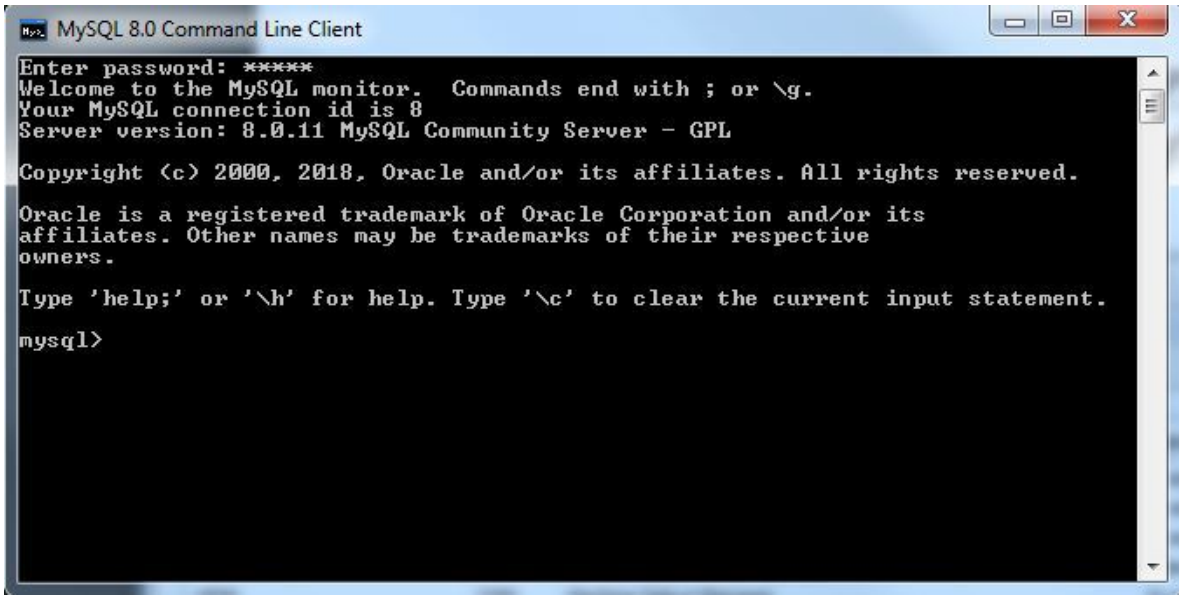
La pagina di installazione proporrà un riepilogo dei prodotti pronti per l'installazione come la seguente:



Alla fine, selezioniamo *Strong Password* per specificare la password dell'utente principale, ti prego di custodire tale password in un luogo sicuro perché verrà spesso utilizzata. Questa password è fondamentale in quanto password dell'utente con i privilegi massimi.

Quando richiesto sarà necessario impostare MySQL come servizio di Windows in modo che parta all'avvio del sistema operativo e non sarà necessario avviarlo manualmente. Proseguite con le impostazioni predefinite e successivamente vi consiglio di selezionare entrambe le voci per eseguire MySQL Workbench e MySQL Shell.

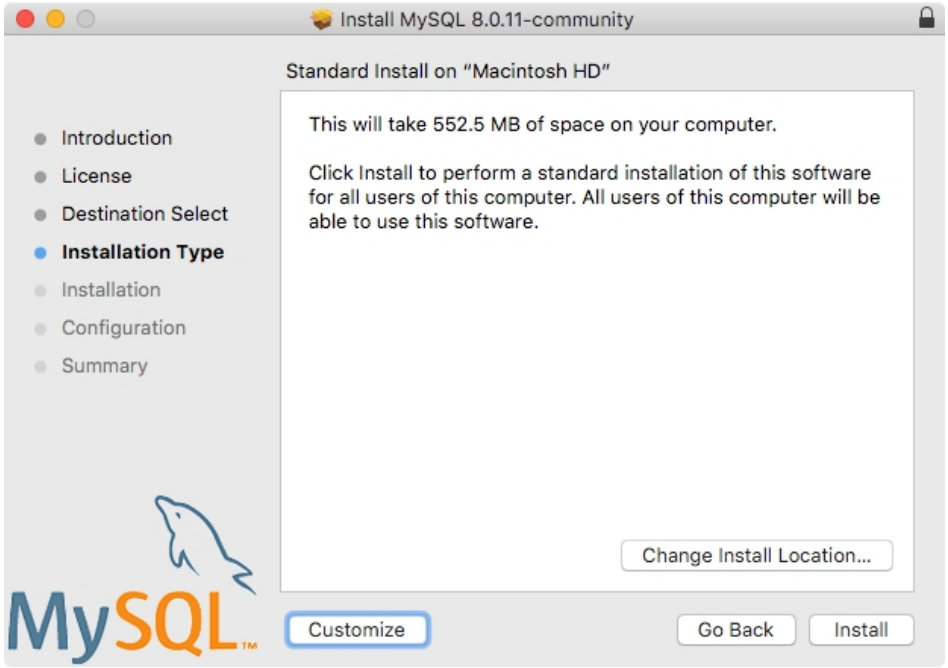
Se l'installazione è andata a buon fine vedrai un terminale simile al seguente:


MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.11 MySQL Community Server - GPL
Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>

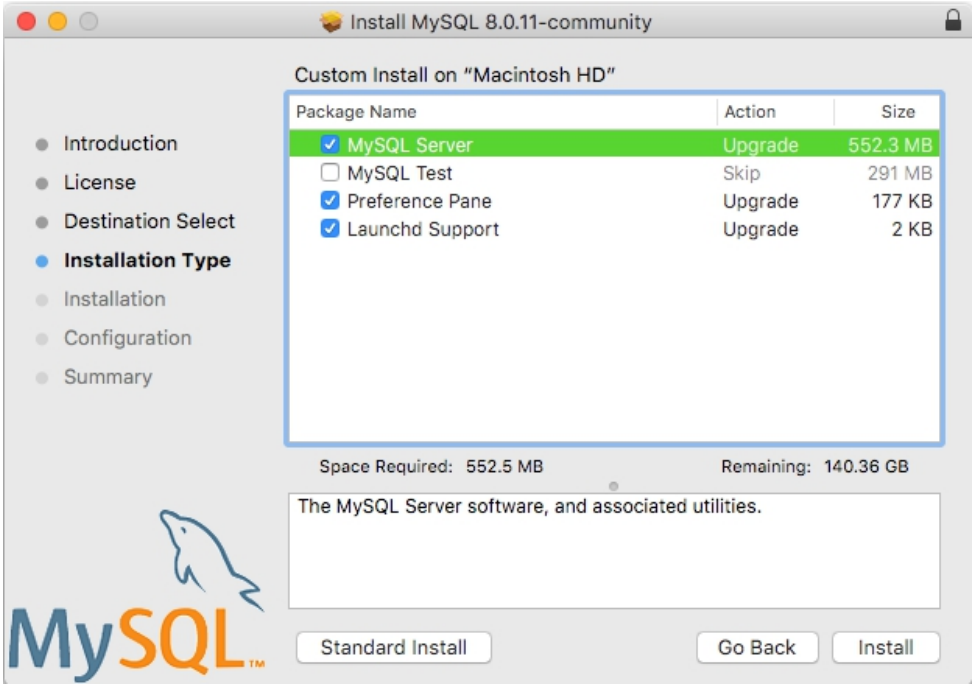
macOS

Puoi scaricare un archivio *.dmg* dal sito ufficiale di MySQL https://dev.mysql.com/downloads/mysql/ e seguire le istruzioni dell'interfaccia utente. Quando il download sarà completato potrai montare il disco immagine con un semplice doppio click per vedere il contenuto di quanto scaricato.

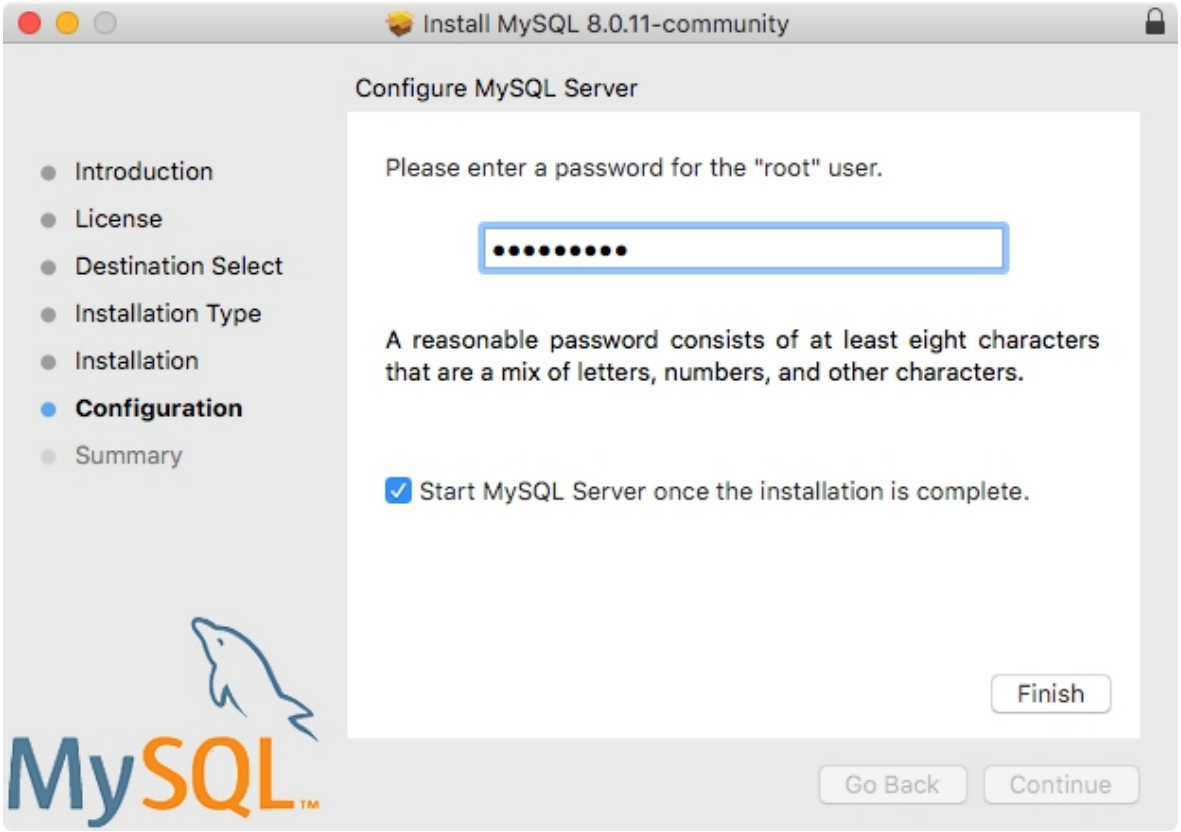
La procedura guidata faciliterà l'installazione del software chiedendo la cartella dove si intende installare MySQL:



Successivamente è necessario selezionare tutte le opzioni come nell'immagine seguente:



Nei passi successivi è necessario definire il tipo di password da usare, selezioniamo *Use Strong Password Encryption* e digitiamo la password per l'utente root ovvero l'utente con i privilegi massimi:



Se l’installazione è andata a buon fine saremo in grado di verificarlo con il comando:

mysql.server status

Se MySQL è avviato potremo entrare nella console tramite il comando:

mysql -u root -p

Il risultato, dopo aver inserito correttamente la password, sarà qualcosa simile alla seguente schermata:

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.11 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Capitolo 4
Comandi SQL

Database

In questo ebook stiamo usando l’utente root per scopi dimostrativi ma è sempre consigliato usare un utente specifico con privilegi specifici su database o tabelle.

Dopo aver effettuato l’accesso alla console di MySQL con il comando:

mysql -u root -p

Il primo passaggio nella gestione dei dati per qualsiasi database è la creazione del database stesso. Questa attività può variare dall'elementare al complicato, a seconda delle tue esigenze. Molti sistemi includono strumenti grafici (come MySQL Workbench) che consentono di creare completamente il database con qualche click. Questa funzione è sicuramente utile per risparmiare tempo, ma dovresti comprendere le istruzioni SQL che vengono eseguite in risposta ai click del mouse. Attraverso l'esperienza personale, abbiamo imparato l'importanza di creare un buon script di installazione SQL. Un file di script contiene il codice SQL necessario per ricostruire completamente uno o più database; lo script spesso include elementi del database come indici, procedure e trigger.

La conoscenza della sintassi SQL è fondamentale in quanto ti consente di applicare le tue conoscenze ad altri sistemi di database relazionali.

La prima considerazione riguarda il livello di autorizzazione infatti è necessario assicurarsi che si disponga delle impostazioni di autorizzazione a livello di amministratore di sistema o che l'amministratore di sistema abbia concesso l'autorizzazione per il comando CREATE DATABASE.

Per iniziare è fondamentale sapere quali sono i database già definiti di default durante l'installazione, MySQL consente di fare ciò tramite il comando:

mysql> show databases;

Probabilmente vedrai soltanto i database utili al funzionamento di MySQL se non ne hai ancora creati. Per creare un database di nome dbTEST sarà sufficiente digitare ed eseguire:

mysql> CREATE DATABASE dbTEST;

Dopo aver creato un database bisogna selezionarlo per i successivi comandi:

mysql> USE dbTEST;

A questo punto siamo pronti per usare il database appena creato, popolandolo di tabelle piene di dati. Prima di riempire le tabelle è importante **normalizzare** il database ovvero suddividere i dati in componenti separati per ridurre la ripetizione dei dati stessi. Esistono diversi livelli di normalizzazione ed ogni livello riduce la ripetizione dei dati. La normalizzazione dei dati può essere un processo estremamente complesso ma esistono numerosi strumenti di progettazione del database che possono esserti d'aiuto.

Ci sono diversi fattori che possono influenzare la progettazione del database, ad esempio lo spazio disponibile sul disco, la velocità con cui viene aggiornato il database o la velocità con cui vengono recuperati i dati.

Lo spazio sul disco è un fattore importante e da tenere in mente infatti anche se siamo in un'era dove gli smartphone possono archiviare Terabyte di dati, ricorda che più grande è il tuo database, più tempo ci vuole per recuperare i record. Se non hai ottimizzato la struttura del DB, è probabile che tu abbia ripetuto inutilmente gran parte dei tuoi dati. Spesso, però, può verificarsi il problema opposto infatti potresti aver cercato di normalizzare completamente la progettazione delle tue tabelle con il database e, in tal modo, hai creato molte tabelle. Anche in questo caso, qualsiasi operazione di

query eseguita su questo database potrebbe richiedere molto tempo per essere eseguita. I database progettati in questo modo sono talvolta difficili da manutenere perché la struttura della tabella potrebbe oscurare l'intento del progettista. Questo problema sottolinea l'importanza di documentare sempre il codice o il design in modo che chi possa lavorare con te possa avere un'idea di cosa stavi pensando nel momento in cui hai creato la struttura del tuo database.

Tabelle

L'obiettivo di progettazione più importante che dovresti avere è quello di creare la struttura della tabella in modo tale che ognuna abbia una chiave primaria ed una chiave esterna. La chiave primaria serve per garantire che:

- Ogni record sia univoco all'interno di una tabella (nessun altro record all'interno della tabella ha tutte le sue colonne uguali a qualsiasi altro);
- i dati in una colonna non siano ripetuti in nessun altro punto della tabella.

Per quanto riguarda il secondo obiettivo, la colonna con dati completamente unici in tutta la tabella è nota come **chiave primaria** . Una chiave esterna è un campo che collega una tabella alla chiave primaria o alla chiave esterna di un'altra tabella. Facciamo un esempio: supponiamo di avere tre tabelle: BOLLETTE, CONTO_CORRENTE e AZIENDA.

La chiave primaria nella tabella BOLLETTE potrebbe essere il campo NOME o, sarebbe meglio, una combinazione di NOME con qualche altro campo. Il campo IBAN nella tabella CONTO_CORRENTE è la chiave primaria per quella tabella mentre il campo NOME è la chiave primaria per la tabella AZIENDA.

Le chiavi esterne in questo esempio sono probabilmente facili da individuare. Il campo IBAN nella tabella BOLLETTE unisce la tabella

BOLLETTE alla tabella CONTO_CORRENTE. Il campo NOME nella tabella BOLLETTE unisce la tabella BOLLETTE alla tabella AZIENDA. Se si trattasse di un progetto di database completo, si otterrebbero molte più tabelle e suddivisioni dei dati. Ad esempio, il campo BANCA nella tabella CONTO_CORRENTE potrebbe puntare ad una tabella contenente tutte le informazioni bancarie come indirizzi e numeri di telefono. La tabella AZIENDA può essere collegata con un'altra tabella (o database) contenente informazioni sulla società e sui suoi prodotti.

CREATE

Vediamo come creare la tabella BOLLETTE in MySQL:

```
mysql> CREATE TABLE BOLLETTE (
    ->  NOME VARCHAR(30),
    ->  IMPORTO FLOAT,
    ->  IBAN VARCHAR(30));
Query OK, 0 rows affected (0.52 sec)
```

Abbiamo creato la tabella che conterrà il campo NOME di una lunghezza massima di 30 caratteri, IMPORTO e l'IBAN del conto corrente (che è univoco).

MySQL consente di identificare ciò che può essere archiviato in una colonna ed un valore NULL può sembrare quasi un ossimoro, perché avere un campo con un valore NULL vuol dire che il campo in realtà non ha alcun valore memorizzato in esso. Quando si crea una tabella, MySQL consente di indicare una colonna con le parole chiave NOT NULL in modo da indicare che la colonna non può contenere valori NULL per nessun record nella tabella. NOT NULL significa che ogni record deve avere un valore effettivo in questa colonna. Aggiorniamo l'esempio precedente con questa clausola:

```
mysql> CREATE TABLE BOLLETTE (
```

```
    -> NOME VARCHAR(30) NOT NULL,
    -> IMPORTO FLOAT,
    -> IBAN VARCHAR(30) NOT NULL);
Query OK, 0 rows affected (0.52 sec)
```

In questa tabella vuoi salvare il nome dell'azienda a cui devi pagare una fattura, insieme all'importo della fattura. Se il campo NOME e/o IBAN non fosse memorizzato, il record sarebbe privo di significato.

INSERT

Adesso che abbiamo la tabella possiamo inserire i dati al suo interno:

mysql> INSERT INTO BOLLETTE VALUES("ENEL",22.5, IT000000001231231');

Query OK, 1 row affected (0.15 sec)

mysql> INSERT INTO BOLLETTE VALUES(NULL, 25000, 'IT000000000012312331');

ERROR 1048 (23000): Column 'NOME' cannot be null

Puoi notare che il secondo record nell'esempio precedente non contiene un valore per il nome. Poiché la tabella è stata creata con NOT NULL per il campo NOME, è stato generato un errore. Una buona regola è che la chiave primaria e tutti i campi della chiave esterna non debbano mai contenere valori NULL.

Chiavi primarie

Uno dei tuoi obiettivi di progettazione dovrebbe essere quello di avere una colonna unica all'interno di ogni tabella. Questa colonna costituisce un campo per la chiave primaria in modo da impedire l'inserimento di valori di campo con chiave duplicati nel database. Dovresti notare diverse cose quando scegli un campo chiave infatti spesso viene usato un campo che viene incrementato per ogni riga aggiunta, il che rende questo campo per impostazione predefinita sempre una chiave univoca. Oltre ad essere davvero utile, è molto più veloce ritrovare un valore intero piuttosto che una stringa di 80 caratteri. Moltiplica questo concetto su centinaia di tabelle ed avrai dimensioni inferiori del database rispettando questa regola.

Ora possiamo creare le tabelle che abbiamo menzionato in precedenza:

```
mysql> CREATE TABLE CONTO_CORRENTE (
    -> IBAN VARCHAR(30) NOT NULL,
    -> SALDO FLOAT,
    -> BANCA VARCHAR(30));
Query OK, 0 rows affected (0.51 sec)

mysql> CREATE TABLE AZIENDA (
    -> NOME VARCHAR(30) NOT NULL,
    -> INDIRIZZO VARCHAR(50),
    -> CITTA VARCHAR(30),
    -> STATO CHAR(2));
Query OK, 0 rows affected (0.74 sec)
```

Nella creazione della tabella ho usato diversi tipi di dati ma balza all'occhio la differenza tra CHAR e VARCHAR. Entrambi consentono di memorizzare delle stringhe di caratteri ma il primo ha una lunghezza fissa mentre il secondo ha lunghezza variabile. Entrambi consentono di definire una lunghezza massima ma con VARCHAR si occupa solo lo spazio necessario. Nel campo indirizzo potremmo memorizzare una stringa di 40 caratteri o di 10, consentendo un notevole risparmio rispetto all'uso di CHAR.

Inseriamo alcuni dati al loro interno:

```
mysql> INSERT INTO CONTO_CORRENTE
VALUES('IT000000000012312331', 105.22, 'BANCA 1');
Query OK, 1 row affected (0.20 sec)

mysql> INSERT INTO AZIENDA VALUES('ENEL', 'VIA PIPPO, 2',
'ROMA','IT');
```

Query OK, 1 row affected (0.34 sec)

Come avrai notato è possibile inserire i comandi SQL su un’unica riga come in questo caso o su più righe. È fondamentale, in entrambi i casi, terminare ogni singola istruzione con un punto e virgola (;) in modo che l’interprete possa capire dove finisce un’istruzione e ne inizia un’altra, soprattutto se si creano degli script con molteplici istruzioni.

ALTER TABLE

Molte volte la progettazione del database cambia perché cambiano i requisiti o magari abbiamo dimenticato qualcosa che va aggiunta. L'istruzione ALTER TABLE consente all'amministratore o al progettista del database di modificare la struttura di una tabella dopo che è stata creata. Il comando ALTER TABLE consente di eseguire sostanzialmente due operazioni:

- Aggiungere una colonna ad una tabella esistente
- Modificare una colonna già esistente
- Cancellare una colonna

La sintassi prevede rispettivamente:

- ALTER TABLE nome_tabella ADD nome_colonna tipo_colonna;
- ALTER TABLE nome_tabella MODIFY COLUMN nome_colonna tipo_colonna;
- ALTER TABLE nome_tabella DROP COLUMN nome_colonna tipo_colonna.

Ad esempio, proviamo a modificare il campo STATO della tabella AZIENDA:

mysql> ALTER TABLE AZIENDA MODIFY COLUMN STATO VARCHAR(3);

Query OK, 1 row affected (1.33 sec)

Records: 1 Duplicates: 0 Warnings: 0

Puoi aumentare o diminuire la lunghezza delle colonne a tuo piacere; comunque non è possibile ridurre la lunghezza di una

colonna se la dimensione di uno dei suoi valori è maggiore del valore che si desidera assegnare alla lunghezza della colonna.

Per l'istruzione ALTER TABLE, tuttavia, esistono alcune restrizioni infatti si può cambiare una colonna da NOT NULL a NULL, ma non viceversa. Una colonna può essere modificata da NULL a NOT NULL solo se la colonna non contiene alcun valore NULL.

DROP TABLE

MySQL fornisce ovviamente un comando per rimuovere completamente una tabella da un database. Il comando DROP TABLE elimina una tabella insieme a tutte le viste e gli indici associati. È necessario prestare attenzione con questo comando perché dopo averlo eseguito, non è più possibile tornare indietro.

L'uso più comune dell'istruzione DROP TABLE è legato alla creazione di tabelle temporanee. Dopo aver completato tutte le operazioni sulla tabella temporanea, si può digitare un’istruzione DROP TABLE simile alla seguente:

```
mysql> DROP TABLE TABELLA_TEMP;
```

DROP DATABASE

Qualora si intenda cancellare un intero database, per esempio, dopo una migrazione avvenuta con successo si può usare il seguente comando:

```
mysql> DROP DATABASE NOME_DB;
```

Capitolo 5
Recuperare i dati

Spesso si parla di **query SQL** e spesso si pensa ad un'interrogazione del database per recuperare dati. In realtà una query SQL può essere un comando per eseguire una delle seguenti operazioni:

- Creare o eliminare una tabella;
- Inserire, modificare o eliminare righe o campi;
- Cercare in diverse tabelle informazioni specifiche e restituire i risultati in un ordine specifico.

La sintassi SQL è abbastanza flessibile e facile da capire, sebbene ci siano regole da seguire come in qualsiasi linguaggio di programmazione. Presta molta attenzione alle maiuscole e alle minuscole, alla spaziatura e alla separazione logica dei componenti di ciascuna query per parole chiave SQL. Scrivere bene delle query aiuterà te e chiunque altro a capire rapidamente cosa stai cercando di fare.

SELECT

Per recuperare i dati dalla tabella creata e popolata nel capitolo precedente puoi usare l'istruzione SELECT:

```
mysql> SELECT * FROM CONTO_CORRENTE;

+---------------------+--------+---------+

| IBAN                | SALDO  | BANCA   |

+---------------------+--------+---------+

| IT000000000012312331 | 105.22 | BANCA 1 |
```

```
+---------------------+--------+---------+

1 row in set (0.01 sec)
```

L'asterisco (*) in SELECT * indica al database di restituire tutte le colonne associate alla tabella indicata descritta dalla clausola FROM. Il database determina l'ordine in cui restituire le colonne. Per specificare l'ordine delle colonne, è possibile digitare qualcosa del tipo:

```
mysql> SELECT BANCA, IBAN, SALDO FROM CONTO_CORRENTE;

+---------+---------------------+--------+

| BANCA   | IBAN                | SALDO  |

+---------+---------------------+--------+

| BANCA 1 | IT000000000012312331 | 105.22 |

+---------+---------------------+--------+

1 row in set (0.00 sec)
```

Devi notare che il nome di ogni colonna è elencato nella clausola SELECT. L'ordine in cui sono elencate le colonne è l'ordine in cui appariranno nell'output. L’elenco delle colonne da mostrare è separato da virgole e la clausola FROM è separata da uno spazio. Se non hai intenzione di mostrare tutte le colonne presenti in una tabella puoi anche limitarti a selezionare una colonna come segue:

```
mysql> SELECT IBAN FROM CONTO_CORRENTE;

+---------------------+

| IBAN                |

+---------------------+
```

```
| IT000000000012312331 |
+----------------------+
1 row in set (0.00 sec)
```

DISTINCT

Talvolta alcuni dati possono ripetersi all’interno di una colonna e, se abbiamo bisogno valori univoci, è necessario usare la parola chiave DISTINCT. Inseriamo altre righe nella tabella BOLLETTE:

```
mysql> INSERT INTO BOLLETTE VALUES('FASTWEB', 35, 'IT000000001231231');

Query OK, 1 row affected (0.20 sec)


mysql> INSERT INTO BOLLETTE VALUES('ENEL', 104.22, 'IT000000003333334');

Query OK, 1 row affected (0.10 sec)


mysql> INSERT INTO BOLLETTE VALUES('FASTWEB', 110, 'IT000000003333334');

Query OK, 1 row affected (0.18 sec)


mysql> SELECT * FROM BOLLETTE;

+------------+---------+-------------------+

| NOME       | IMPORTO | IBAN              |

+------------+---------+-------------------+

| Amazon AWS |    22.5 | IT000000001231231 |

| FASTWEB    |      35 | IT000000001231231 |

| ENEL       |  104.22 | IT000000003333334 |

| FASTWEB    |     110 | IT000000003333334 |

+------------+---------+-------------------+
```

4 rows in set (0.00 sec)

In questo caso abbiamo aggiunto le bollette dell'ufficio che paghiamo da un altro conto corrente, quindi, con un codice IBAN diverso. Come puoi vedere nella SELECT vengono restituiti tutti i valori ma in questo caso vogliamo una lista di codici IBAN senza duplicati:

```
mysql> SELECT DISTINCT IBAN FROM BOLLETTE;

+-------------------+

| IBAN              |

+-------------------+

| IT000000001231231 |

| IT000000003333334 |

+-------------------+
```

2 rows in set (0.12 sec)

Se vuoi trovare un determinato articolo o gruppo di elementi nel tuo database, hai bisogno di una o più condizioni. Le condizioni sono contenute nella clausola WHERE, ad esempio cerchiamo tutte le bollette pagate relative a Fastweb:

```
mysql> SELECT * FROM BOLLETTE WHERE NOME='FASTWEB';

+---------+---------+-------------------+

| NOME    | IMPORTO | IBAN              |

+---------+---------+-------------------+

| FASTWEB |      35 | IT000000001231231 |

| FASTWEB |     110 | IT000000003333334 |

+---------+---------+-------------------+
```

2 rows in set (0.00 sec)

SELECT, FROM e WHERE sono le tre clausole più utilizzate in MySQL. WHERE semplicemente fa in modo che le tue query siano più selettive infatti non usandola, la cosa più utile che potresti fare con una query, è visualizzare tutti i record nella tabella selezionata.

Capitolo 6
Gli operatori

Gli operatori sono gli elementi usati all'interno di un'espressione per articolare il modo in cui desideri che determinate condizioni recuperino i dati. Gli operatori si dividono in diversi gruppi: aritmetici, di confronto, per caratteri, logici ecc.

Aritmetici

Gli operatori aritmetici sono addizione (+), sottrazione (-), moltiplicazione (*), divisione (/) e modulo (%). I primi quattro sono semplici da capire e li conosciamo tutti. L'operatore modulo, invece, restituisce il resto intero di una divisione.

È possibile usare questi operatori nella selezione dei dati dalle tabelle. Supponiamo di dover aggiungere ad ogni valore nella tabella BOLLETTE il costo per effettuare il bonifico che è di € 1,50:

```
mysql> SELECT NOME, IMPORTO, IMPORTO+1.5 FROM BOLLETTE;

+---------+---------+--------------------+

| NOME    | IMPORTO | IMPORTO+1.5        |

+---------+---------+--------------------+

| ENEL    |    22.5 |                 24 |

| FASTWEB |      35 |               36.5 |

| ENEL    |  104.22 |                105.72 |

| FASTWEB |     110 |              111.5 |

+---------+---------+--------------------+

4 rows in set (0.00 sec)
```

In questo caso abbiamo aggiunto il costo del bonifico e MySQL assume come intestazione della colonna IMPORTO+1.5. Il nome di questa colonna potrebbe essere difficile da usare o ricordare, tuttavia, possiamo assegnare un nome diverso tramite un **alias** . Un alias viene definito tramite la parola chiave AS quindi proviamo a rinominare la colonna IMPORTO TOTALE:

```
mysql> SELECT NOME, IMPORTO, IMPORTO+1.5 AS 'IMPORTO TOTALE' FROM BOLLETTE;

+---------+---------+--------------------+

| NOME    | IMPORTO | IMPORTO TOTALE     |

+---------+---------+--------------------+

| ENEL    |    22.5 |                24 |

| FASTWEB |      35 |              36.5 |

| ENEL    |  104.22 |                105.72 |

| FASTWEB |     110 |              111.5 |

+---------+---------+--------------------+

4 rows in set (0.00 sec)
```

Attenzione! Tutto quello che vedi nella tabella è frutto di un’elaborazione successiva al recupero dei dati infatti se recuperi i dati della tabella vedrai che conterrà i dati originali. In sostanza, abbiamo creato una colonna virtuale o derivata, modificando i valori di una colonna esistente. Lo stesso può essere fatto per sottrazione, moltiplicazione e divisione:

```
mysql> SELECT NOME, IMPORTO, IMPORTO-1 AS 'IMPORTO SCONTATO' FROM BOLLETTE;

+---------+---------+--------------------+

| NOME    | IMPORTO | IMPORTO SCONTATO   |
```

```
+---------+---------+-------------------+
| ENEL    |    22.5 |              21.5 |
| FASTWEB |      35 |                34 |
| ENEL    |  104.22 | 103.22000122070312 |
| FASTWEB |     110 |               109 |
+---------+---------+-------------------+
4 rows in set (0.00 sec)
```

mysql> SELECT NOME,IMPORTO, IMPORTO-(IMPORTO*0.1) AS 'SCONTATO del 10%' FROM BOLLETTE;

```
+---------+---------+------------------+
| NOME    | IMPORTO | SCONTATO del 10% |
+---------+---------+------------------+
| ENEL    |    22.5 |            20.25 |
| FASTWEB |      35 |             31.5 |
| ENEL    |  104.22 | 93.79800109863281 |
| FASTWEB |     110 |               99 |
+---------+---------+------------------+
4 rows in set (0.00 sec)
```

mysql> SELECT NOME,IMPORTO, IMPORTO/2 AS 'SCONTATO del 50%' FROM BOLLETTE;

```
+---------+---------+------------------+
| NOME    | IMPORTO | SCONTATO del 50% |
```

```
+---------+---------+-------------------+
| ENEL    |    22.5 |             11.25 |
| FASTWEB |      35 |              17.5 |
| ENEL    |  104.22 | 52.11000061035156 |
| FASTWEB |     110 |                55 |
+---------+---------+-------------------+
4 rows in set (0.00 sec)
```

Infine, l'operatore modulo restituisce il resto di una divisione perciò supponiamo di dividere 176 caramelle tra 6 bambini:

```
mysql> SELECT 176%6 AS 'CARAMELLE RESTANTI';
+--------------------+
| CARAMELLE RESTANTI |
+--------------------+
|                  2 |
+--------------------+
1 row in set (0.00 sec)
```

Confronto

Fedeli al loro nome, questi operatori confrontano le espressioni e restituiscono un valore: TRUE, FALSE o NULL cioè vero, falso o sconosciuto. Qui bisogna prestare attenzione perché MySQL si comporta in modo inaspettato con i valori NULL. Confronti del tipo valore = NULL o valore <> NULL producono sempre un risultato NULL (che non è né vero né falso) perché non è possibile decidere se veri o falsi. Di conseguenza anche NULL = NULL restituisce

NULL perché non è possibile determinare se un valore sconosciuto è uguale ad un altro valore sconosciuto. Per verificare davvero se un valore è NULL, bisogna utilizzare gli operatori IS NULL e IS NOT NULL.

```
mysql> SELECT NOME, IMPORTO FROM BOLLETTE WHERE IMPORTO IS NULL;

Empty set (0.00 sec)


mysql> SELECT NOME,IMPORTO FROM BOLLETTE WHERE IMPORTO IS NOT NULL;

+---------+---------+

| NOME    | IMPORTO |

+---------+---------+

| ENEL    |    22.5 |

| FASTWEB |      35 |

| ENEL    |  104.22 |

| FASTWEB |     110 |

+---------+---------+

4 rows in set (0.00 sec)
```

Gli operatori di confronto sono sei:

- Maggiore (>)
- Maggiore o uguale (>=)
- Minore (<)
- Minore o uguale (<=)
- Uguale (=)
- Diverso (<>)

Facciamo qualche esempio con i dati inseriti nella tabella BOLLETTE:

```
mysql> SELECT NOME, IMPORTO FROM BOLLETTE WHERE IMPORTO < 40;

+---------+---------+

| NOME    | IMPORTO |

+---------+---------+

| ENEL    |    22.5 |

| FASTWEB |      35 |

+---------+---------+

2 rows in set (0.00 sec)
```

```
mysql> SELECT NOME, IMPORTO FROM BOLLETTE WHERE IMPORTO >= 35;

+---------+---------+

| NOME    | IMPORTO |

+---------+---------+

| FASTWEB |      35 |

| ENEL    |  104.22 |

| FASTWEB |     110 |

+---------+---------+

3 rows in set (0.00 sec)
```

```
mysql> SELECT NOME, IMPORTO FROM BOLLETTE WHERE IMPORTO = 110;
```

```
+---------+---------+
| NOME    | IMPORTO |
+---------+---------+
| FASTWEB |     110 |
+---------+---------+
1 row in set (0.00 sec)
```

```
mysql> SELECT NOME, IMPORTO FROM BOLLETTE WHERE IMPORTO <> 110;
+---------+---------+
| NOME    | IMPORTO |
+---------+---------+
| ENEL    |    22.5 |
| FASTWEB |      35 |
| ENEL    |  104.22 |
+---------+---------+
3 rows in set (0.00 sec)
```

Caratteri

È possibile utilizzare gli operatori dei caratteri per manipolare il modo in cui sono rappresentate le stringhe di caratteri, sia nell'output dei dati sia nel processo di posizionamento delle condizioni sui dati da recuperare. Se volessi selezionare parti di un database che si adattano ad un pattern ma non corrispondono esattamente? È possibile utilizzare il segno uguale ed eseguire tutti i casi possibili, ma tale processo sarebbe noioso e richiederebbe molto tempo.

La parola chiave in questo caso è LIKE:

```
mysql> SELECT NOME FROM BOLLETTE WHERE NOME LIKE 'FA%';

+---------+

| NOME    |

+---------+

| FASTWEB |

| FASTWEB |

+---------+

2 rows in set (0.00 sec)
```

In questo caso abbiamo selezionato tutti i nomi che iniziano per “FA” perciò il database restituisce soltanto due righe. Se utilizzato all'interno di un'espressione LIKE, % è un carattere jolly. Puoi spostare il carattere jolly all’inizio della stringa in modo da cercare tutti i campi che terminano per “FA”:

```
mysql> SELECT NOME FROM BOLLETTE WHERE NOME LIKE '%FA';

Empty set (0.00 sec)
```

Un altro modo per cercare tra le stringhe, limitandoci ad un singolo carattere come jolly, consiste nell’usare il trattino basso (_):

```
mysql> SELECT DISTINCT NOME FROM BOLLETTE WHERE NOME LIKE 'ENE_';

+------+

| NOME |

+------+

| ENEL |

+------+

1 row in set (0.00 sec)
```

Puoi anche combinare i due elementi in modo da cercare tutte le stringhe che contengono una E seguita da una sola lettera ma preceduta da altre lettere:

```
mysql> SELECT NOME FROM BOLLETTE WHERE NOME LIKE '%E_';

+---------+

| NOME    |

+---------+

| ENEL    |

| FASTWEB |

| ENEL    |

| FASTWEB |

+---------+

4 rows in set (0.00 sec)
```

Logici

Gli operatori logici separano due o più condizioni nella clausola WHERE di un'istruzione SQL. Vogliamo cercare una bolletta da pagare, sappiamo che è di Fastweb e che ammonta a 35 euro. Possiamo usare l'operatore logico AND per unire queste condizioni in una WHERE:

```
mysql> SELECT * FROM BOLLETTE WHERE NOME = 'FASTWEB' AND IMPORTO = 35;

+---------+---------+-------------------+

| NOME    | IMPORTO | IBAN              |

+---------+---------+-------------------+

| FASTWEB |      35 | IT000000001231231 |

+---------+---------+-------------------+

1 row in set (0.00 sec)
```

AND indica che le espressioni su entrambi i lati devono essere vere per restituire TRUE. Se una delle espressioni è falsa, AND restituisce FALSE.

Puoi anche usare OR per riassumere una serie di condizioni. Se uno dei confronti è vero, OR restituisce TRUE:

```
mysql> SELECT * FROM BOLLETTE WHERE NOME = 'FASTWEB' OR IMPORTO < 40;

+---------+---------+-------------------+

| NOME    | IMPORTO | IBAN              |

+---------+---------+-------------------+

| ENEL    |    22.5 | IT000000001231231 |
```

| FASTWEB | 35 | IT000000001231231 |

| FASTWEB | 110 | IT000000003333334 |

+---------+---------+-------------------+

3 rows in set (0.00 sec)

Come vedi per questa query abbiamo un risultato composto da 3 righe perché vengono unite quelle con nome pari a “FASTWEB” con quelle che hanno importo inferiore a 40 euro.

Un altro operatore spesso utilizzato è il NOT che nega un’intera espressione. Spesso viene utilizzato per mantenere una certa leggibilità del codice dato che si potrebbe usare l’operatore <> per verificare la disuguaglianza. Se la condizione a cui viene applicato è TRUE, NOT la rende FALSE. Se la condizione dopo l’operatore NOT è FALSE, diventa TRUE:

mysql> SELECT NOME FROM BOLLETTE WHERE NOME NOT LIKE '%EL';

+---------+

| NOME |

+---------+

| FASTWEB |

| FASTWEB |

+---------+

2 rows in set (0.00 sec)

Insiemistici

Le tabelle possono essere viste come degli insiemi di elementi perciò esistono dei comandi SQL che ti consentono di unire le righe di due tabelle come se fossero degli insiemi. Uno di questi comandi è UNION che restituisce i risultati di due query tranne le righe duplicate:

mysql> SELECT NOME FROM AZIENDA UNION SELECT NOME FROM BOLLETTE;

```
+---------+
| NOME    |
+---------+
| AMAZON  |
| ENEL    |
| FASTWEB |
+---------+
3 rows in set (0.00 sec)
```

Questo comando è molto utile quando si hanno valori duplicati o ripetuti in diverse tabelle perché magari importate da altri database o altri sistemi. UNION ALL ha lo stesso funzionamento di UNION, tranne per il fatto che non elimina i duplicati.

Un altro modo per selezionare gli elementi contenuti in un altro insieme consiste nell'usare la parola chiave IN:

mysql> SELECT * FROM BOLLETTE WHERE NOME IN ('ENEL', 'TEST');

```
+------+---------+------------------+
| NOME | IMPORTO | IBAN             |
+------+---------+------------------+
```

```
| ENEL |    22.5 | IT000000001231231 |

| ENEL |  104.22 | IT000000003333334 |

+------+---------+-------------------+

2 rows in set (0.00 sec)
```

La query appena scritta equivale ad una clausola WHERE con più condizioni in OR quindi corrisponde a:

```
mysql> SELECT * FROM BOLLETTE WHERE NOME = 'ENEL' OR NOME = 'TEST';

+------+---------+-------------------+

| NOME | IMPORTO | IBAN              |

+------+---------+-------------------+

| ENEL |    22.5 | IT000000001231231 |

| ENEL |  104.22 | IT000000003333334 |

+------+---------+-------------------+

2 rows in set (0.00 sec)
```

In questo caso risulta molto più leggibile e facile da usare la clausola IN rispetto a diverse condizioni in OR.

Un’altra clausola può essere molto utile quando si vuole selezionare una serie di valori compresi tra due estremi (inclusi). Con la parola chiave BETWEEN si indicano tutti gli elementi compresi tra il valore a sinistra (estremo inferiore) e l’elemento a destra (estremo superiore) della clausola:

```
mysql> SELECT * FROM BOLLETTE WHERE IMPORTO BETWEEN 22 AND 105;

+---------+---------+-------------------+
```

```
| NOME    | IMPORTO | IBAN              |
+---------+---------+-------------------+
| ENEL    |    22.5 | IT000000001231231 |
| FASTWEB |      35 | IT000000001231231 |
| ENEL    |  104.22 | IT000000003333334 |
+---------+---------+-------------------+
3 rows in set (0.00 sec)
```

Capitolo 7
Funzioni

Le funzioni in SQL consentono di eseguire delle azioni come determinare la somma di una colonna o la conversione in maiuscolo di tutti i caratteri di una stringa.

Esistono diversi tipi di funzioni:

- Funzioni aggregate
- Funzioni per data e ora
- Funzioni aritmetiche
- Funzioni per i caratteri
- Funzioni di conversione
- Funzioni varie

Le funzioni aumentano notevolmente la tua capacità di manipolare le informazioni recuperate utilizzando le funzioni di base di SQL descritte in precedenza.

Funzioni aggregate

Le prime cinque funzioni aggregate, COUNT, SUM, AVG, MAX e MIN, definite nello standard ANSI.

Le funzioni di aggregazione sono anche chiamate funzioni di gruppo e restituiscono un valore basato sui valori in una colonna. La funzione COUNT restituisce il numero di righe che soddisfano una condizione nella clausola WHERE. Potresti anche voler sapere quante righe sono contenute nella tua tabella quindi digiterai:

mysql> SELECT COUNT(*) AS 'NUMERO BOLLETTE' FROM BOLLETTE;

+----------+

| NUMERO BOLLETTE |

```
+----------------+
|              4 |
+----------------+
1 row in set (0.00 sec)
```

Mentre per sapere quante bollette sono di Fastweb la query sarà:

```
mysql> SELECT COUNT(*) AS 'BOLLETTE FASTWEB' FROM BOLLETTE WHERE NOME = 'FASTWEB';
+-----------------+
| BOLLETTE FASTWEB |
+-----------------+
|              2  |
+-----------------+
1 row in set (0.12 sec)
```

La funziona SUM addiziona tutti i valori in una colonna, per esempio se volessimo calcolare il totale di tutte le bollette:

```
mysql> SELECT SUM(IMPORTO) AS TOTALE FROM BOLLETTE;
+------------------+
| TOTALE           |
+------------------+
| 271.7200012207031 |
+------------------+
1 row in set (0.00 sec)
```

La funzione SUM prende in input solo valori numerici quindi non restituisce alcun risultato sui caratteri. Inoltre, puoi usare le condizioni WHERE per filtrare le righe da sommare.

La funzione AVG consente di creare la media dei valori contenuti in una colonna. Ad esempio, la media delle nostre bollette sarà pari a:

```
mysql> SELECT AVG(IMPORTO) AS MEDIA FROM BOLLETTE;

+------------------+

| MEDIA            |

+------------------+

| 67.93000030517578 |

+------------------+

1 row in set (0.00 sec)
```

Molto semplici ed intuitive sono le funzioni MAX e MIN che consentono di recuperare rispettivamente il valore massimo ed il valore minimo contenuti in una colonna:

```
mysql> SELECT MAX(IMPORTO) AS 'IMPORTO MASSIMO' FROM BOLLETTE;

+----------------+

| IMPORTO MASSIMO |

+----------------+

|            110 |

+----------------+

1 row in set (0.00 sec)
```

```
mysql> SELECT MIN(IMPORTO) AS 'IMPORTO MINIMO' FROM BOLLETTE;

+----------------+

| IMPORTO MINIMO |

+----------------+

|          22.5 |

+----------------+

1 row in set (0.00 sec)
```

A fini statistici possono aiutare anche la varianza e la deviazione standard. La varianza fornisce una misura della variabilità dei valori assunti dalla variabile stessa; in particolare, la misura di quanto essi si discostino in modo quadratico rispetto alla media aritmetica o del valore atteso. La deviazione standard, invece, è un indice di dispersione statistico, vale a dire una stima della variabilità di una popolazione di dati. Useremo la parola chiave VARIANCE per calcolare la varianza e STD_DEV per calcolare la deviazione standard:

```
mysql> SELECT VARIANCE(IMPORTO) AS 'VARIANZA' FROM BOLLETTE;

+--------------------+

| VARIANZA           |

+--------------------+

| 1558.7797221496585 |

+--------------------+

1 row in set (0.00 sec)
```

mysql> SELECT STDDEV(IMPORTO) AS 'DEVIAZIONE STANDARD' FROM BOLLETTE;

```
+---------------------+
| DEVIAZIONE STANDARD |
+---------------------+
|   39.48138450142875 |
+---------------------+
1 row in set (0.00 sec)
```

Funzioni per data e ora

Spesso è fondamentale usare delle date in un database per memorizzare il giorno, il mese e l'anno di un evento oppure per memorizzare l'orario e la data di accesso di un utente.

La funzione ADDDATE consente di aggiungere un determinato valore ad una determinata unità temporale, ad esempio giorni, mesi, anni ecc.

mysql> SELECT NOW() AS ADESSO;

```
+---------------------+
| ADESSO              |
+---------------------+
| 2020-06-27 12:48:27 |
+---------------------+
1 row in set (0.00 sec)
```

mysql> SELECT ADDDATE(NOW(),INTERVAL 1 MONTH) AS 'TRA UN MESE';

```
+--------------------+
| TRA UN MESE        |
+--------------------+
| 2020-07-27 12:48:35 |
+--------------------+
1 row in set (0.00 sec)
```

I valori che possibile specificare sono molti ma i più usati sono:

- MICROSECOND
- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH
- YEAR
- MINUTE_SECOND
- HOUR_MINUTE
- DAY_HOUR
- YEAR_MONTH

Come puoi notare sono tutti abbastanza semplici da capire ma gli ultimi possono trarre in inganno. Se volessimo aggiungere ore e minuti oppure minuti e secondi ad una data, non serve invocare due volte la funzione. Useremo rispettivamente HOUR_MINUTE e MINUTE_SECOND per risparmiare tempo:

```
mysql> SELECT ADDDATE(NOW(),INTERVAL '1:30' HOUR_MINUTE) AS 'TRA UN\'ORA E MEZZO';
```

```
+--------------------+
| TRA UN'ORA E MEZZO  |
+--------------------+
| 2020-06-27 14:24:32 |
+--------------------+
1 row in set (0.00 sec)
```

Personalmente dimentico sempre quanti giorni hanno i mesi e, se non fosse per la famosa filastrocca, non saprei proprio come fare. Possiamo usare la funzione LAST_DAY() per recuperare l'ultimo giorno di un mese:

```
mysql> SELECT LAST_DAY(NOW());
+-----------------+
| LAST_DAY(NOW()) |
+-----------------+
| 2020-06-30      |
+-----------------+
1 row in set (0.01 sec)
```

MySQL è davvero utile con le date, infatti, dando in input la tua data di nascita può calcolare quanti giorni fa sei nato. Non ci credi? Verifichiamo insieme:

```
mysql> SELECT DATEDIFF(NOW(), "1980-05-11") AS 'QUANTI GIORNI FA\' SONO NATO?';
+------------------------------+
| QUANTI GIORNI FA' SONO NATO? |
```

```
+-----------------------------+
|                       14657 |
+-----------------------------+
1 row in set (0.00 sec)
```

Immagina quanto è potente MySQL e quanto può semplificarti la vita. Potresti fare lo stesso calcolo nel tuo linguaggio di programmazione ma probabilmente avresti bisogno di più di una riga.

Funzioni aritmetiche

Quando si parla di funzioni si pensa sempre alle funzioni matematiche come valore assoluto, seno, coseno e radice quadrata. Tutte queste funzioni e molte altre sono disponibili in MySQL, vediamo come usarle:

```
mysql> SELECT ABS(-10);
+----------+
| ABS(-10) |
+----------+
|       10 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT SIN(0);
+--------+
| SIN(0) |
```

```
+--------+
|      0 |
+--------+
1 row in set (0.00 sec)

mysql> SELECT COS(0);
+--------+
| COS(0) |
+--------+
|      1 |
+--------+
1 row in set (0.00 sec)
mysql> SELECT SQRT(16);
+----------+
| SQRT(16) |
+----------+
|        4 |
+----------+
1 row in set (0.00 sec)
```

Funzioni per caratteri

Per i caratteri ci sono davvero tante funzioni disponibili infatti possiamo concatenare stringhe, rendere i caratteri di una stringa tutti minuscoli o maiuscoli, eseguire un riempimento a sinistra o a destra fino a raggiungere una determinata lunghezza e tanto altro.

Per raggiungere gli obiettivi appena descritti useremo rispettivamente le funzioni CONCAT, LOWER, UPPER, LPAD, RPAD.

```
mysql> SELECT CONCAT('test', 'concat');

+-------------------------+

| CONCAT('test', 'concat') |

+-------------------------+

| testconcat              |

+-------------------------+

1 row in set (0.00 sec)


mysql> SELECT LOWER('TESTdiLOwer');

+---------------------+

| LOWER('TESTdiLOwer') |

+---------------------+

| testdilower         |

+---------------------+

1 row in set (0.12 sec)


mysql> SELECT UPPER('TESTdiLOwer');
```

```
+---------------------+
| UPPER('TESTdiLOwer') |
+---------------------+
| TESTDILOWER          |
+---------------------+
1 row in set (0.00 sec)
mysql> SELECT LPAD('TEST', 10, "-");
+----------------------+
| LPAD('TEST', 10, "-") |
+----------------------+
| ------TEST           |
+----------------------+
1 row in set (0.00 sec)

mysql> SELECT RPAD('TEST', 10, "-");
+----------------------+
| RPAD('TEST', 10, "-") |
+----------------------+
| TEST------           |
+----------------------+
1 row in set (0.00 sec)
```

Un’altra funzione molto usata è TRIM che consente di eliminare gli spazi ad inizio e fine stringa. È possibile anche eliminare solo gli

spazi ad inizio stringa usando LTRIM e solo a fine stringa usando RTRIM.

```
mysql> SELECT LTRIM('   |a|');

+----------------+

| LTRIM('   |a|') |

+----------------+

| |a|            |

+----------------+

1 row in set (0.00 sec)


mysql> SELECT RTRIM('   |a|    ');

+--------------------+

| RTRIM('   |a|    ') |

+--------------------+

|    |a|             |

+--------------------+

1 row in set (0.00 sec)


mysql> SELECT TRIM('   |a|    ');

+-------------------+

| TRIM('   |a|    ') |

+-------------------+

| |a|               |

+-------------------+
```

1 row in set (0.12 sec)

La funziona SUBSTR (abbreviazione di substring) accetta tre argomenti e consente di estrarre delle parti da una stringa di input. Il primo argomento indica la stringa da cui estrarre, il secondo argomento è la posizione del primo carattere da considerare mentre il terzo argomento è il numero di caratteri da mostrare.

```
mysql> SELECT SUBSTR('PIPPO', 3);
+--------------------+
| SUBSTR('PIPPO', 3) |
+--------------------+
| PPO                |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT SUBSTR('PIPPO', 3, 2);
+-----------------------+
| SUBSTR('PIPPO', 3, 2) |
+-----------------------+
| PP                    |
+-----------------------+
1 row in set (0.00 sec)
```

Infine, ma non meno importante, troviamo la funzione LENGTH che restituisce il numero di caratteri contenuti in una stringa:

```
mysql> SELECT LENGTH('PIPPO');
```

```
+----------------+
| LENGTH('PIPPO') |
+----------------+
|               5 |
+----------------+
1 row in set (0.00 sec)
```

Capitolo 8
JOIN delle tabelle

Una delle funzionalità più potenti di MySQL è la sua capacità di raccogliere e manipolare dati da più tabelle. Senza questa funzione dovresti archiviare tutti gli elementi di dati necessari per ogni applicazione in una tabella. Senza la possibilità di selezionare i dati da più tabelle sarebbe necessario archiviare gli stessi dati in più tabelle. Immagina di dover riprogettare, ricostruire e ripopolare tabelle e database ogni volta hai bisogno di una query con una nuova informazione. L'istruzione JOIN di MySQL consente di progettare tabelle più piccole, più specifiche e più facili da gestire rispetto alle tabelle più grandi.

Il modo più semplice per unire delle tabelle consiste nell'unirle in una FROM. In questo modo si effettua un prodotto cartesiano ovvero si avrà una tabella di NxM righe dove N sono le righe della prima tabella, M le righe della seconda. Il prodotto cartesiano può essere creato unendo le due tabelle nella clausola FROM, separandole dalla virgola o, in alternativa, con una CROSS JOIN.

```
mysql> SELECT * FROM BOLLETTE,CONTO_CORRENTE;

+---------+---------+------------------+------------------+--------+---------+

| NOME    | IMPORTO | IBAN             | IBAN             | SALDO  | BANCA   |

+---------+---------+------------------+------------------+--------+---------+

| ENEL    |    22.5 | IT000000001231231 | IT000000001231231 | 105.22 | BANCA 1 |

| FASTWEB |      35 | IT000000001231231 | IT000000001231231 | 105.22 | BANCA 1 |

| ENEL    |  104.22 | IT000000003333334 | IT000000001231231 | 105.22 | BANCA 1 |
```

| FASTWEB | 110 | IT000000003333334 | IT000000001231231 | 105.22 | BANCA 1 |

+---------+---------+-----------------+-----------------+--------+---------+

4 rows in set (0.00 sec)

mysql> SELECT * FROM BOLLETTE CROSS JOIN CONTO_CORRENTE;

+---------+---------+-----------------+-----------------+--------+---------+

| NOME | IMPORTO | IBAN | IBAN | SALDO | BANCA |

+---------+---------+-----------------+-----------------+--------+---------+

| ENEL | 22.5 | IT000000001231231 | IT000000001231231 | 105.22 | BANCA 1 |

| FASTWEB | 35 | IT000000001231231 | IT000000001231231 | 105.22 | BANCA 1 |

| ENEL | 104.22 | IT000000003333334 | IT000000001231231 | 105.22 | BANCA 1 |

| FASTWEB | 110 | IT000000003333334 | IT000000001231231 | 105.22 | BANCA 1 |

+---------+---------+-----------------+-----------------+--------+---------+

4 rows in set (0.00 sec)

Come vedi ogni riga della tabella BOLLETTE è legata ad ogni riga della tabella CONTO_CORRENTE. Ti ricordo che finora nella prima tabella ci sono solo 4 righe, nella seconda esiste solo 1 riga.

INNER JOIN

In MySQL una INNER JOIN seleziona tutte le righe da due tabelle per mostrare un risultato se e solo se entrambe le tabelle soddisfano le condizioni specificate nella clausola ON. La clausola ON specifica su quali campi si vuole restringere la selezione.

```
mysql> SELECT * FROM CONTO_CORRENTE CC INNER JOIN BOLLETTE B ON CC.IBAN = B.IBAN;

+------------------+--------+---------+---------+---------+------------------+

| IBAN             | SALDO  | BANCA   | NOME    | IMPORTO | IBAN             |

+------------------+--------+---------+---------+---------+------------------+

| IT000000001231231 | 105.22 | BANCA 1 | ENEL    |    22.5 | IT000000001231231 |

| IT000000001231231 | 105.22 | BANCA 1 | FASTWEB |      35 | IT000000001231231 |

+------------------+--------+---------+---------+---------+------------------+

2 rows in set (0.00 sec)
```

In questo caso abbiamo effettuato una INNER JOIN sul campo IBAN che è presente in entrambe le tabelle.

Inoltre, abbiamo dato un alias alle tabelle in modo da poter capire su quale stiamo operando con maggiore facilità, in particolare, la tabella CONTO_CORRENTE è stata ridenominata CC mentre BOLLETTE è stata ridenominata B.

LEFT JOIN

Se pensiamo due tabelle come due insiemi che contengono elementi (ovvero righe) possiamo unire i due insiemi considerando la parte in comune oltre a tutti gli elementi della tabella a sinistra.

mysql> SELECT * FROM CONTO_CORRENTE CC LEFT JOIN BOLLETTE B ON CC.IBAN = B.IBAN;

```
+------------------+--------+---------+---------+---------+------------------+
| IBAN             | SALDO  | BANCA   | NOME    | IMPORTO | IBAN             |
+------------------+--------+---------+---------+---------+------------------+
| IT000000001231231 | 105.22 | BANCA 1 | ENEL    |    22.5 | IT000000001231231 |
| IT000000001231231 | 105.22 | BANCA 1 | FASTWEB |      35 | IT000000001231231 |
+------------------+--------+---------+---------+---------+------------------+
```

2 rows in set (0.00 sec)

RIGHT JOIN

Proprio come nella LEFT JOIN, nella RIGHT JOIN il concetto è uguale ma cambia il verso infatti si recuperano tutti gli elementi in comune oltre a quelli della tabella di destra.

mysql> SELECT * FROM CONTO_CORRENTE CC RIGHT JOIN BOLLETTE B ON CC.IBAN = B.IBAN;

```
+------------------+--------+---------+---------+---------+------------------+
| IBAN             | SALDO  | BANCA   | NOME    | IMPORTO | IBAN             |
+------------------+--------+---------+---------+---------+------------------+
| IT000000001231231 | 105.22 | BANCA 1 | ENEL    |    22.5 | IT000000001231231 |
```

| IT000000001231231 | 105.22 | BANCA 1 | FASTWEB | 35 | IT000000001231231 |

| NULL | NULL | NULL | ENEL | 104.22 | IT000000003333334 |

| NULL | NULL | NULL | FASTWEB | 110 | IT000000003333334 |

+----------+--------+---------+---------+---------+----------+

4 rows in set (0.00 sec)

SELF JOIN

È anche possibile creare qualsiasi tipo di JOIN sulla stessa tabella, infatti, stiamo per creare una INNER JOIN della tabella BOLLETTE con sé stessa:

```
mysql> SELECT CONCAT(BB.NOME, ' - ', BB.IMPORTO) AS 'NOME - IMPORTO'
    -> FROM BOLLETTE B
    -> INNER JOIN BOLLETTE BB ON
    -> BB.NOME = B.NOME
    -> WHERE BB.NOME = 'ENEL';
+----------+
| NOME - IMPORTO |
+----------+
| ENEL - 22.5    |
| ENEL - 22.5    |
```

```
| ENEL - 104.22  |

| ENEL - 104.22  |

+----------------+

4 rows in set (0.00 sec)
```

In questo caso abbiamo creato una INNER JOIN della tabella con sé stessa assegnando un alias (B e BB) per distinguerle. Puoi sostituire alla INNER JOIN una LEFT JOIN o una RIGHT JOIN per recuperare quello di cui hai bisogno dalla tua tabella. Forse ti stai chiedendo a cosa serve esattamente la SELF JOIN. Effettivamente in questo caso avremmo potuto selezionare le righe con una semplice condizione WHERE ma non tutte le tabelle sono così concise e soprattutto i dati contenuti, a volte, sono molto più complessi. Immagina di “ereditare” il database creato e popolato da qualcun altro e scopri che ogni tabella contiene almeno 15 colonne. In questo caso una SELF JOIN potrebbe davvero tornarti utile.

Conclusioni

In questo ebook abbiamo cercato di mostrarti la flessibilità e la potenza di MySQL, spiegando come applicare queste funzionalità ai problemi del mondo reale. Avrai imparato a maneggiare funzioni, query, tabelle e database perciò sforzati per rendere il tuo codice più leggibile ed evita query troppo lunghe o troppo complesse. Molti dei concetti e delle funzioni che abbiamo analizzato sono uguali o simili in altri database infatti puoi usarle con DB2, PostgreSQL e tanti altri. Dovresti avere una chiara comprensione di quali siano le parole chiave, come e quali dati possono essere memorizzati, come strutturare un database e come recuperare informazioni utili da esso. Hai incontrato dei problemi o MySQL ti restituisce degli errori? Non scoraggiarti, spesso si tratta di errori di sintassi quindi basterà consultare la documentazione ufficiale per risolvere il problema. Se imposti le tue query su più righe ti basterà leggere bene il messaggio di MySQL per notare su quale riga si trova l'errore. Non sarai mai immune da errori di sintassi o errori logici, ma acquisendo più esperienza con MySQL, imparerai come evitare molti problemi. Fondamentalmente, gli errori possono essere eccellenti opportunità di apprendimento infatti "sbagliando s'impara".